

NATIONAL BUREAU OF STANDARDS REPORT

10 859

FINAL REPORT
NBS PROJECT WHERE
in support of
USAAMCA PROJECT MNPLS

by
Applied Mathematics Division
and
Computer Services Division



U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards¹ was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Center for Computer Sciences and Technology, and the Office for Information Programs.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of a Center for Radiation Research, an Office of Measurement Services and the following divisions:

Applied Mathematics—Electricity—Heat—Mechanics—Optical Physics—Linac Radiation²—Nuclear Radiation²—Applied Radiation²—Quantum Electronics³—Electromagnetics³—Time and Frequency³—Laboratory Astrophysics³—Cryogenics³.

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials and the following divisions:

Analytical Chemistry—Polymers—Metallurgy—Inorganic Materials—Reactor Radiation—Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services to promote the use of available technology and to facilitate technological innovation in industry and Government; cooperates with public and private organizations leading to the development of technological standards (including mandatory safety standards), codes and methods of test; and provides technical advice and services to Government agencies upon request. The Institute also monitors NBS engineering standards activities and provides liaison between NBS and national and international engineering standards bodies. The Institute consists of the following divisions and offices:

Engineering Standards Services—Weights and Measures—Invention and Innovation—Product Evaluation Technology—Building Research—Electronic Technology—Technical Analysis—Measurement Engineering—Office of Fire Programs.

THE CENTER FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus within the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Center consists of the following offices and divisions:

Information Processing Standards—Computer Information—Computer Services—Systems Development—Information Processing Technology.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS and other agencies of the Federal Government; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world, and directs the public information activities of the Bureau. The Office consists of the following organizational units:

Office of Standard Reference Data—Office of Technical Information and Publications—Library—Office of International Relations.

¹ Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.

² Part of the Center for Radiation Research.

³ Located at Boulder, Colorado 80302.

NATIONAL BUREAU OF STANDARDS REPORT

NBS PROJECT

6300505
2050505

May 31, 1972

NBS REPORT

10 859

FINAL REPORT
NBS PROJECT WHERE
in support of
USAAMCA PROJECT MNPLS

by
Applied Mathematics Division
and
Computer Services Division

IMPORTANT NOTICE

NATIONAL BUREAU OF STANDARDS
for use within the Government. Before
and review. For this reason, the
whole or in part, is not authorized
Bureau of Standards, Washington,
the Report has been specifically prepared

Approved for public release by the
Director of the National Institute of
Standards and Technology (NIST)
on October 9, 2015.

accounting documents intended
subjected to additional evaluation
listing of this Report, either in
Office of the Director, National
the Government agency for which
lies for its own use.



U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

List of Contents

1. Executive summary of report
2. Description of general approach used in this report
3. Review of analyses to date
4. Conclusions
5. Appendixes

Working paper no. 6: Error propagation for one-dimensional location
of two units. By M.H. Pearl

Working paper no. 7: Algorithms. By D.J. Sookne

Working paper no. 8: LS, LSS and relocation. By J.A. Lechner

Working paper no. 9: The MNPLS simulation ("WHERE SIM").
By R.H.F. Jackson

1. Executive summary of report

This report summarizes the work done at the National Bureau of Standards (NBS) under its project WHERE in support of the US Army Advanced Materiel Concepts Agency's (USAAMCA) project MNPLS (Micro Navigation and Position Location System).

MNPLS is a system that utilizes a time-division multiplexed frequency that is shared by n units, all of which are synchronized by a suitable electromagnetic signal during each time interval ΔT . Each unit is assigned one or more time slots within ΔT during which it emits an electromagnetic signal. This signal in turn is received by several but not necessarily all other units in the field which measure the time of flight of the EM signal; from the knowledge of the time and the assignment of the time slots for each unit, the distance between the sending unit and the receiving unit can be inferred.

For NBS the problem to be solved was twofold:

(1) To determine the feasibility of such a system in the presence of measurement errors. If all measurements were exact three units in the plane or 4 units in three-space would suffice to locate another (or all other) units. In the presence of error more measurements are needed and the "best" solution that minimizes total error in some mathematical sense has to be found. The fundamental problem for NBS was to show that the overall error does not increase unduly as the system develops in time.

(2) To develop a computer simulation for such a system that allows the investigation of real-time scenarios with a sufficiently large number of units undergoing battle-field manoeuvres, each with prescribed movements over suitably long times. The simulation program determines the distance between any two units i and j , perturbs that distance according to suitable assumptions about the measurement errors, and supplies to the master computer (the program that determines positions from the distance measurements) the distances as though they came from actual field measurements.

The output of the program consists of the successive locations of all units, the errors between true position and calculated position, and a measure of confidence derived from the available measurements but not from the knowledge of exact position (which is only available to the simulation program but not to the position location algorithm).

NBS has designed a set of mathematical algorithms for position location determination that allow the determination of successive positions of n field units, from inputs containing only distances between units, or perhaps those distances accompanied by independent measurements of differences in height above sea level. The experiments based on actual deployments in the Boston area show that, with a suitable number of fixed units, the average error of position over all units can be kept well within acceptable limits. Based on a range-error standard derivation of 6m , and using all 83 units as reporters, the maximum error during 2 hours of actual time remained under 110m. The average error was 4.5m over all units, with only 3 units "lost" more than 8 times out of 240 tries. When only the 39 fixed units were used as reporters, the average error was 4.7m, the maximum was 40m, but 15 units were "lost" more than 50 times out of 240. (The term "locators" refers to those users that do actually receive a signal from the locatee, and do report a time-of-flight to the master computer; the term "reporters" refers to those users that have been instructed to report time-of-flight measurements to the computer. Because of "line-of-sight" requirements, not all of the latter may be able to serve as locators for any given location-operation.)

A simulation package WHERSM was designed that simulates the operation of a Micro-Navigation and Position Location System. This program allows one to stage a wide variety of movements for any battle-field scenario; it generates the inter-unit distances, perturbs the ranges according to agreed upon error distributions and provides the links to the position location algorithms which were also developed at NBS for this purpose.

The position location algorithms fall into three distinct categories:

(1) Two-dimensional algorithms. Where the terrain is flat, no altitude reference is necessary and the resulting algorithms become two-dimensional.

The sum $\sum w_i (r_i^2 - d_i^2)^2$ is minimized, where the sum extends over some or all reporting units and the weights are determined by the estimates of variance of each of the reported distances d_i .

(2) Pseudo three-dimensional algorithms. Where the altitudes are so small in relation to the horizontal distances that the slant distances are nearly the same as the horizontal distances it becomes necessary to reduce the position location problem to the two-dimensional one above. Not only is the three-dimensional problem ill defined mathematically, but we have also a genuine multiplicity of solutions if all locators are nearly in a plane and trying to locate a unit outside the plane. The determination of heights that are small against the horizontal distance is impossible within any reasonable limits of accuracy and the altitude must therefore be determined either by barometric reference or a reference to a digitized terrain map. For example, at a distance of 1km, a range uncertainty of only 6m leads to a height uncertainty of 110m.

(3) Fully three-dimensional algorithms are being used when the altitudes are comparable to the horizontal distances. Again, a penalty-function approach is used that minimizes $\sum w_i (d_i^2 - r_i^2)^2$ with suitably determined weights depending on the variance of the measurement and the sum being taken over some or all reporting units.

The following results were obtained:

- o The test runs conducted on the Boston deployment show that stability of the position-location process is indeed achievable as long as some units remain fixed.

- o Computer size for position location:

Runs were all carried out on the UNIVAC 1108 at NBS. For this machine, the formula for computing the number of computer words required by the position-location algorithms as they are currently programmed is as follows:

$$26N + 7R + 3874,$$

where N = number of units in the field, and R = the maximum number of ranges being reported.

As a matter of possible interest, the number of additional words required by the simulation program (into which the position-location algorithms were inserted) is:

$$91N + N_S + 4002,$$

where N_S = the number of subcycles in a cycle of the MNPLS (see Working Paper No. 9) . Furthermore, the preprocessor to the simulation requires 39,714 computer words.

- o Timing for position location operation:

Several runs are available from which the time required by the algorithms can be estimated. One run was done using LSS(3d), with 20 locators;

it required approximately 16.5ms per location operation. A run with 15 locators using slant range reduction took 8-12ms ; with 20 locators, the computation took 14 to 18ms. For the Boston deployment, using 39 reporters (but probably less than 15 locators on the average because of intervisibility restrictions), 8 to 14ms were required; using 83 reporters, 17 to 25ms were required. Since the algorithms are not optimized for running time, it is reasonable to expect some decrease in these times for the operational system.

2. Description of general approach used in this report.

(1) Simulation.

The simulation of the battlefield scenario is done in our NBS-developed computer program WHERSM. This program provides for a "real life" framework for MNPLS through its capability for moving all field units along prescribed paths, or subjecting these to random changes in their direction of movement. WHERSM generates all the inter-unit distances(ranges); checks, if necessary, for intervisibility (by look up in an intervisibility matrix) ; disturbs the exact ranges according to known error distribution laws; and transmits these "measurements" to the position location algorithms that we developed here at NBS and which are described below.

The simulation package also provides a facility for monitoring the operation of such a system under a variety of movement scenarios; and ultimately it provides for a variety of outputs that are desired for both checking out the feasibility of the entire system and what might be called the ultimate output that would be useful to the field commander who is interested in the whereabouts of his units. For details of WHERSM, the reader is referred to working paper No. 9 in the appendix of this report.

6

(2) Algorithms.

The basic LSS algorithm that we finally used after numerous experiments designed to eliminate competing ones is the so-called "least-squares squared" algorithm, which minimizes the penalty function

$$E = \sum_i w_i (d_i^2 - r_i^2)^2$$

where i is taken over all of the locators and w_i is a weight that increases with decreasing variance of the observation r_i , the reported distance of the locatee from the locator with index i . The quantity d_i is the calculated distance between (x_i, y_i, z_i) and (x, y, z) the estimated position of the locatee. The solution (x, y, z) is obtained iteratively from the equations $\partial E / \partial x = 0$, $\partial E / \partial y = 0$, $\partial E / \partial z = 0$ once a starting point is known.

As long as a trajectory is known for a given locatee, the initial guess is simply obtained from extrapolation. If the unit is lost, however, a set of linear equations is solved (LSL method). Details of the LSL method can be found in NBS report 10663: First Interim Progress Report on NBS Project WHERE in support of USA AMCA Project MNPLS.

The appendix contains a list of variables used in the simulation code. Since this is the part of the NBS system that interfaces with the outside world, we believe that those are the only variables that have to be listed. The variables in the algorithm parts are internal and are thus not referencable from outside.

3. Review of analyses to date.

3.1. General.

As originally conceived, the testing was to comprise two phases: initial testing to choose the best position-location algorithm, and final verification testing to determine how well the chosen algorithm would work on the Boston deployment. This neat plan did not fit the real-life flow of events, however: every stone turned over revealed two more yet unturned. The process of developing a new system is bound to contain such loops. For example, one does not know how to optimize the algorithm until one knows the system design, but the system design is sensitive to the effectiveness of the chosen algorithm. Thus the two phases gradually blurred into each other. Furthermore, in the limited time allotted for this study, it seemed that effort would be better spent getting reasonable answers to more of the real questions, rather than "optimizing" on a smaller subset of the questions while assuming particular (and questionable) answers to the others.

For these reasons, a strictly chronological account of our analyses would not be very helpful. Instead, it seems reasonable to describe the test results by first building a framework of the questions to be answered, and then describing the tests performed and the answers obtained. The next section consists of a set of questions; the following section presents the answers to date, as gleaned from the experimental program and from certain theoretical studies undertaken specifically to support that program.

3.2. Questions pertinent to the MNPL system.

The general question that constitutes the focus of a feasibility study is: Will the proposed system perform satisfactorily under the conditions of

its intended use? This question is however a very broad one; it asks essentially whether the final outcome of the feasibility study is "yes", "no" or "maybe", and thus does not give much guidance in conducting the study. Thus one is led to pose narrower queries:

a. General Questions

- (1) Under highly optimistic conditions, will the system work? (If not, terminate the study.)
- (2) Under what reasonable near-minimum conditions can the system be made to work?
- (3) How well does it work?

b. More Specific Questions

The next question leads to a whole set of sub-questions, which are listed along with the main one.

- (4) What position-location algorithm(s) should be used? For each candidate algorithm, consider:
 - (a) Computer time required?
 - (b) One-step accuracy?
 - (c) Error propagation (sensitivity to locators' position errors)?
 - (d) Sensitivity to occasional very large errors in the range measurements?
 - (e) Affected by dimensionality?
 - (f) Need an initial estimate of position?

The next set of questions refers to restrictions placed on the units in the system.

- (5) How many (if any) units must remain fixed?
- (6) Must they be permanently fixed in known positions?
- (7) How many reporters are required, to keep the errors small?
- (8) Will ground units have to carry altimeters? Instead, will a digitized map, stored in the computer, suffice?
- (9) How accurate must altimeters be (both ground and airborne)?

Questions (10) - (19) refer to design of the algorithm.

- (10) How often should units report?
- (11) How many aircraft are required, in order to use a three-dimensional algorithm on ground units?
- (12) How should one handle near-planarity (when any aircraft present are flying very low)?
- (13) When should the algorithm:
 - (a) ignore height differences?
 - (b) reduce slant ranges to horizontal distances?
 - (c) use a 3-dimensional method?
 - (d) use 3d for aircraft, slant range reduction for ground units?
- (14) How should the locators be chosen?
- (15) How weight the locators?
 - (a) What is the distribution of range errors?
 - (b) How handle the occasional occurrence of range measurements with extremely large errors?
 - (c) How estimate the accuracy of an estimated position?
- (16) Should the locators be "relocated" ? (i.e., should the locators' positions be adjusted, based on discrepancies between measured ranges and calculated ranges to the best estimate of locatee's position?)

- (17) Use only fixed units as reporters? How soon after they become fixed?
- (18) Can the algorithm determine (satisfactorily) which units are fixed?
If not, could a moving/stopped indicator bit be included in the transmission?
- (19) After the system has been running for a while, and errors have accumulated, is it possible to "restart" the system ---i.e. to treat the set of (currently) fixed units in toto, simultaneously, allowing all position coordinates to vary, in order to converge to good estimates for their positions?

3.3. Conduct, progress, and results of the test program.

The first few questions, above, suggest a beginning emphasis on favorable conditions. The rationale is that, when very little is known about a system, more is learned faster by starting with conditions where the system works tolerably well, and then investigating the effects of departures from these conditions. Accordingly, the first major project was to devise, develop, program, and test several candidate algorithms for determining the position of a locatee, using the (perturbed) ranges to a set of locators, and assuming the locators' positions to be known exactly. That is, attention was focussed first on questions(4) . Some of these algorithms were suitable for 2 or 3 dimensions, others were usable in 2 dimensions only. Methods considered were: Linear Method (LM), Smallest Tangent Circle (STC), Least Squares Linear (LSL), Minmax, Least Squares(LS), and Least Squares Squared (LSS). The first two methods use exactly 3 locators in 2 dimensions, or 4 locators in 3 dimensions. The next two are generalizations of the first two,

respectively, to an arbitrary number of locators. Least squares and LSS are penalty-function methods. All are fully described in earlier sections, in Working Papers Nos. 7 and 8, and/or in NBS Report 10663, First Interim Progress Report on Project WHERE in support of USAAMCA Project MNPLS.

Tests were first conducted in two dimensions. STC turned out considerably more accurate than LM, but slower; LS and LSS agreed closely, and were more accurate than any others, but slower than all but Minmax. Since Minmax was no more accurate, and several times more time-consuming, than LSS, it was dropped from further consideration. Also, since LM was inferior to STC, it was dropped. Thus the survivors at this point were: STC, LSL, LS, and LSS.

The next sequence of tests was directed to question (4)(c). For these runs, errors were applied to the positions of the locators, corresponding to a point in time after the system has been operating long enough to accumulate such errors. In these runs, STC fell down markedly. LS and LSS did better than LSL, but the latter was kept since it alone (of the three methods still in contention) does not need a starting value — i.e., is not an iterative method — and therefore can provide a position estimate when the locatee first joins the net or has been out of touch for some time.

Three-dimensional tests confirmed the pattern of results above, and also accented the importance of geometry: when the locators are nearly coplanar (collinear in 2 dimensions), LSL often produces a very poor estimate, because the planes (lines) that it tries to fit are nearly parallel. LS and LSS also have problems, but not as severe as LSL. These problems were

accentuated in 3d, simply because locators on the surface of the earth are relatively coplanar; the corresponding situation in 2d would have the locators strung out nearly on a single line, a less likely situation.

In terms of computer time LSL was best, with LSS next. Since LSS was devised to perform like LS but faster, these two were then compared for accuracy. 72 trials were run on each method, with 18 users scattered over an 8km by 16km area, with altitudes to 9km, for each of two different values of σ (the standard deviation of the range error). Each of the 18 units served as locatee 4 times for each method and each value of σ . The results were as follows: For $\sigma = 1m$, the position estimates were in error by as much as 3.32m, but the difference between the LS estimate and the LSS estimate was never more than 0.0004m. For $\sigma = 10m$, the errors reached 31.9m, but the two methods gave estimates that differed by no more than 0.045m. As a result of this test, it was concluded that LS and LSS could be used interchangeably so far as accuracy is concerned. Since LSS was considerably faster, LS was dropped. The final result thus was: Use LSS; if no starting value (initial estimate of position) is available, use LSL to obtain one. (Complications develop when there is a choice between 2- and 3-dimensional versions, as we will see.)

The only part of Question (4) yet unanswered is (4)(d): sensitivity to occasional large errors (outliers) in the measured ranges. This is a severe problem for STC and Minmax, since STC uses only three ranges and Minmax concentrates on minimizing the maximum discrepancy. For LSL, LS, and LSS, the matter is treated in a statistical sense, as is commonly done in

fitting regression curves. The true position is estimated, and then the discrepancies between measured and calculated ranges are examined. Any discrepancies significantly larger than the average are attributed to faulty measurements. These measurements are ignored, and the cycle repeated. This approach is sure to detect all really bad errors, provided they occur infrequently, unless there is exactly the minimum number of ranges necessary to produce an estimate. It was been shown to be quite efficient, in the sense that the occasional rejection of measurements "apparently" but not really contaminated with large extraneous errors does not appreciably degrade the accuracy of the estimation procedure. The actual cutoff point for rejection is a variable that will need further investigation as part of a system development effort.

Questions 5,6, and 7 pertain to the set of reporters. (The term "locators" refers to those users that do actually receive a signal from the locatee, and do report a time-of-flight to the master computer; the term "reporters" refers to those users that have been instructed to report time-of-flight measurements to the computer. Due to the requirement for a "line-of-sight", not all of the latter may be able to serve as locators for any given location-operation.) In order to obtain broad(rather than situation-specific) answers to these questions, it was necessary to set up a number of different scenarios, with different numbers of users and different proportions stopped. This diversity was created by employing several kinds of "randomness" . Users were assigned to randomly chosen starting points within a specified rectangle, with aircraft heights specified or randomly chosen;

sequences of azimuths were chosen randomly over a 90° range, and azimuth change times were drawn from an exponential distribution. Stopping and starting times were drawn from exponential distributions, chosen to achieve specified values for the expected (long-term average) proportion of users stopped at any given instant, and for the mean times spent in "fixed" status and in "moving". Intervisibilities were similarly randomized, to achieve specified values for the average time two users maintain communication and the average time they remain out of communication. These features were used in various combinations to investigate most of the remaining questions. The results of those investigations are presented below.

If all the reporters are moving, strange things can happen. Small (random) errors in locating a given reporter propagate into the location of other reporters, so that each remains well-located relative to the others but the whole cluster is displaced (and perhaps rotated) far from its true position. This effect is reasonably well understood, and has been clearly demonstrated. One case involved 30 users, moving more or less in the same direction. 12 users were reporting, with perfect intervisibility (2 dimensions). By the time 100 location operations had been done on each, they had moved 0.9 to 4.1km. The estimated position of the cluster as a whole was 3.3km from its true location, but each element was located within 150m of its true position relative to the others. (Range errors were less than 2m in this case.) The problem can be seen clearly in the following simple example: If all of a collection of fixed users suddenly begin to move, each with the same velocity vector, or all of a collection of users moving with a common

velocity vector suddenly stop, no algorithm based on inter-unit ranges can see the change, because the inter-unit ranges don't change. Therefore, the algorithm will leave the users fixed in the first case, and will continue to move them in the second. This is a special case, of course, but the same principle applies to the average velocity vector when the units begin to move with different velocities - or more generally, to the average change in the velocity vector when they (more or less) all change direction at about the same time.

These results made it plain that one needs some users fixed. The next question was, how many?

Several runs were made with different sets of fixed users, under two conditions: first, that the computer does not know which users are fixed, and second, that it does know (and can use) this information. The first run, with range errors distributed uniformly from -3m to +3m, involved 15 motorized and 15 foot units; in each set of 15, 5 were fixed and reporting, 5 were moving and reporting, and the remaining 5 were moving but not reporting. After 220 cycles (109 minutes), the xy errors ranged from 500m to 10,240m, and were distributed into all four quadrants (more specifically, 14 users were placed NW of their true locations, 5 SW, 10 SE, and 1 NE). For the second run, the subcycle Δt was reduced by a factor of 5, to see if locating the users more often would improve the accuracy. After 613 cycles (60 min), the errors ranged from 570m to 22000m. (The errors grew more slowly per cycle, but faster in real time.) Next, 4 of the fixed users were assumed "known" - i.e., no estimation of their positions was allowed - to provide an

anchor of sorts. After 239 cycles (116 min.), the average error over all users was 9.4m , and the maximum was 23m ; over all cycles, the average was 7.2m , the maximum 82m . But there is some evidence that the errors fluctuate considerably with time: in the set of ten cycles prior to the above (i.e., cycles 221-230), the average was 23m ; the maximum 69m !

At this point, relocation (Question 16) was tried. (Refer to Working Paper No. 8 for technical details.) Tentative conclusions from the dozen or so runs are: In the beginning, relocation increases the errors. Eventually, the errors without relocation become larger than those with relocation; however, this did not show up until about 2 hours of real time had been simulated, by which time the fast-moving users had separated from the slow-moving and fixed users by about 20km . (At this time, the overall average and maximum errors were 19m and 557m with relocation, and 34m and 1036m without. Least Squares was also tried; the average errors were comparable, but the maxima were 570 and 1695 respectively.)

Perhaps more instructive were the shorter runs with differing values of the fraction mentioned in Working Paper No. 8, which measures the extent to which relocation is carried. Runs were done with this fraction set at 0.5, 0.2, 0.1, and 0 (i.e., no relocation). The results showed steady improvement as the fraction went down, with smallest errors occurring when no relocation was done. This shows clearly that there is no advantage to relocation for the first 100 cycles (50 minutes) at least. Still open (and reasonable) is the possibility that relocation should be instituted for a time, now and then, with a small fraction, to help remove accumulated errors.

However, if there are sufficiently many fixed reporters, one can probably do better by periodically going into a survey mode wherein all the fixed units' positions are simultaneously adjusted, effectively "restarting" the system.

At this point, it seemed clear that relocation was not worth its price (more than doubling the algorithm's computer time). This conclusion was checked once more, in the next set of runs, under more realistic conditions, and was supported by the results: the errors were smaller (at 100 cycles) without relocation.

Parallel efforts by ECOM had meanwhile produced a "best engineering judgment" range-error distribution, which can be adequately modeled as "Gaussian with variance 36 m^2 ; mean 0, except that a random 1% of measurements are inflated by about 7 meters." There will also be occasional extremely large errors; a simple preprocessor can weed most of these out in the field system, by simply checking whether the range measurement is compatible with the last known position and the velocity potential of the user. This error distribution was implemented in the balance of our tests. Also implemented were moving boundaries, which can be set to advance at a speed appropriate for foot-soldiers; any motorized unit which reaches the boundary will be reflected off it, so that the motorized units do not stray far from the foot units. Additional features implemented at this time were: a moving/stopped indicator; calculation of weights which reflect the estimated accuracy of a user's estimated position; the ability to choose as locators those users with greatest weights, or those which have been stopped

for a certain number of location-operations; improvement of estimates for the positions of fixed users, by combining successive estimates; a functional representation of terrain; use of an independent measure of the z-coordinate (the height), with random errors; algorithms to reduce slant-range distances to planar (xy-plane) distances, using the independent measure of height difference between locator and locatee; and tests to determine whether a particular location-operation should be treated by slant-range reduction or by a 3-dimensional method. Again, these features were used in various combinations for the runs recounted below.

Before these remaining runs are described, the rational used to "weight" the various range measurements will be explained. No claim is made that these weights are optimal, but it is felt that they are reasonably close to the best values, which is sufficient for a feasibility study. Suppose half the locators had the same x-coordinate as the locatee, and half had the same y-coordinate. Then (assuming ranges to be large relative to range errors) the first half of the locators is useless in determining the x-coordinate, and the second half is useless in determining the y-coordinate. Thus if there are n locators, $n/2$ are used for each coordinate, and so we have $n/2$ estimates (assumed independent) of each of the two coordinates of the locatee's position. Consider for definiteness the x-coordinate. If the locators' position errors are unbiased (i.e., have mean value equal to zero), the best estimate of the locatee's x-coordinate is a weighted average of these $n/2$ values: weighted by the reciprocals of the variances of these observations. Since the observational error is the sum of the locator's x-coordinate error

and the range measurement error, its variance is the sum of the position error variance (determined when the locator was last located) and the measurement error variance ($= 36 \text{ m}^2$). Thus one can calculate an estimate of the x-coordinate error variance, which in turn is used to calculate the weight for this user when he serves as a locator.

[There are two considerations which should be mentioned here. First, locators are not in general split so nicely along perpendicular lines. Does this matter? Of course it does, in the following sense: if more locators serve to estimate x than y , then x is likely to be better known - i.e., to have smaller variance - and y will be less well known. On the average, however, things even out: if azimuths (from locatee to locators) are uniformly distributed from 0 to 360° , the average (or expected) variance in a given direction turns out to be the value derived above. The second consideration is: What if the estimated variances for the locators are wrong? Two consequences follow: (a) To the extent that the variances are different multiples of the assumed values, the estimate of position is less than optimum (because the true relative variances should be used to get the estimate); and to the extent that the variances are (as a group) larger (or smaller) than assumed, the estimated variance of the locatee's position will be too small (or large, respectively). Point (a) is not likely to be important, since it would take very large discrepancies to affect the estimate noticeably. Point (b), on the other hand, should at least be investigated. One technique is to act as though the true variances are proportional to the estimated values, with an unknown constant of proportionality (say c). Then c can be estimated, and used to produce a fair variance estimate. Specifically, if each

range observation has variance $c\sigma_i^2$ (σ_i^2 = location variance + range measurement variance), then the best estimate for the locatee can be found without considering c . If d_i represents the discrepancy between the calculated range from this position to the assumed position of the i -th locator and the corresponding measured range, then $\left[\frac{1}{n-2} \sum d_i^2 / \sigma_i^2 \right]$ is the (multiplicative) correction to be applied to the variance estimate already given. (If $c \sim 1$, this value should also be ~ 1 .) This is a recent result and the corresponding modification has not yet been made to the algorithms.]

A new series of 2-dimensional runs was done, with 30 users. An initial set of fixed users (reporters) was specified. From then on, units became reporters as soon as they had been fixed for ten location-operations, and ceased to be reporters as soon as they began to move. Average times moving and fixed were set at 15 min., so that about 15 (half the units) were stopped at any one time. A $2\frac{1}{2}$ -hour (300 cycle) run was done, and resulted in an overall average error of 3.2m, and a maximum error of 35m; in the last set of 5 cycles, the average was 3.5m, the maximum 11m. Next, a run using longer cycles (90 sec. instead of 30) produced errors of 3.8 and 50.5 overall, 5.7 and 33.8 for the last 5 cycles. Next, the average times fixed and moving were set at 45 min. and this run repeated (because with 15 min. times and $1\frac{1}{2}$ min. cycles, many units didn't stay stopped long enough to be much use as locators). The error figures then came back down toward those of the first run: 3.4 and 33.3 overall, 5.9 and 22.5 for the last 5 cycles.

At this point, the cycle time was returned to 30 sec., and a relocation run was tried, with fraction 0.2. It did poorly: errors 5.8 average and

35 maximum overall, with 8.4 and 17.5 for the last 5 cycles. All errors were in one quadrant, indicating that some instability had been introduced. Next, a run with fraction .001 was done. Since this fraction is so small, the magnitudes of the actual relocations will be essentially negligible; thus the effect of relocation in this run is simply to allow more iterations for the algorithms. (Remember that the number of iterations has been kept small to keep the running time down, with the thought that range errors of the order of 6m don't justify too much precision in locating the true solution of the least squares problem.) Only 100 cycles were done, with results equal (up to roundoff effects) to those without relocation. This supports the conclusion that generally, it is not worth doing too many iterations. (Of course, how many is "too many" depends very much on how good the algorithm is!)

The algorithm was modified to treat aircraft with a 3-dimensional method, while continuing to treat ground units 2-dimensionally. (This is possible since only ground units were being used as locators.) Ten aircraft were added, at (random) altitudes ranging from 128 to 2220m. That low-flying aircraft presented a problem became very obvious, since the algorithm located them where they belonged at some times, near the ground at others, and below ground (mirror image) at still other times. One more attempt was made in this series, using only ranges that were at most 10 times height, to ensure that not all lines of sight were flat. This helped for the second-lowest aircraft, at 339m: it had an average (x,y) error of 19m, with a maximum of 113m. However, the lowest aircraft generally did not have enough acceptable (short enough) ranges to estimate its location. It was decided, therefore, to do a separate systematic

study of aircraft location errors. This will be discussed below. The conclusion is already clear, however, that some independent height information will be required to locate low-flying aircraft, unless other aircraft are serving as locators: however, the use of aircraft as locators will probably require very stable aircraft and good tracking techniques.

A functional representation of terrain - i.e., $z = f(x,y)$, with $f(x,y)$ chosen to give a reasonable surface - was implemented at this stage, to test the use of independent pressure references (p.r.'s) as a source of height-difference information. After consultation with AMCA personnel, these were modeled as giving height with an error that had zero mean and $\sigma = 14\text{m}$ (so that the difference of 2 heights will have $\sigma = 20\text{m}$). 40 units were used, on a terrain that varied 200m in elevation, with 15 units stopped initially; again only units that had been stopped for 10 location-operations were used as locators. The x,y coordinates were obtained by reducing slant ranges to horizontal distances and applying 2d algorithms. In $2\frac{1}{2}$ hours, the overall average error was 3.5m, the maximum 29.2 ; for the last 5 cycles, the average was 4.8, the maximum 13.7 . Next the p.r. σ was set to 0 , with results identical to another run where the terrain was flat and no heights were considered. For extraneous reasons, it only went 260 cycles instead of 300; it had errors of 3.2 (avg), 28.2 (max) overall; 3.3, 14.6 for the last 5 cycles. Next, the height estimates were set at 0 : i.e., the hills and valleys were ignored by the algorithm. The errors were slightly smaller than the ordinary run using p.r.'s: 3.35 and 28.2 overall, 4.4 and 14.3 for the last 5 cycles. This is not surprising, since the errors introduced by the

p.r.'s are comparable to the terrain differences. If p.r.'s don't perform any better, perhaps they should be ignored when only ground units are concerned. However, in particular situations, the terrain heights may introduce systematic errors if ignored, which may then propagate; the p.r. errors at least have the advantage of being random. More investigation of this possibility should be undertaken when more is known about p.r.'s . For the next run, terrain roughness was tripled: with the p.r.'s the errors were essentially the same as on the flatter terrain.

Only two sets of runs are yet to be described: the aircraft error study and the runs on the Boston terrain. The Boston scenario had no aircraft, so it was treated using slant-range reduction with p.r.'s. Two runs using the 1st Brigade constitute the final trials: one with only the fixed users reporting, and one with all units reporting. In preliminary trials, the effect of dropping those reporters which are in another brigade was seen to be minimal, except for a few front-line users which had severe intervisibility problems. The errors for the two primary runs are summarized in Table 1 .

Several points deserve to be mentioned with regard to these results. It will be noted that for 30 of the 44 units, the average error is less when all units serve as reporters. Furthermore, in 10 of the remaining 14, the higher average using all units seems to be due to poor estimates where otherwise (with only the fixed units reporting) there would be no estimates. For most of the units with maximum error above 20m, those maxima occurred with less than 2 fixed locators (i.e., where no estimate would be available using only fixed locators). The greatest error occurred for unit 28, and was 108m ; it

(as well as the two unsuccessful location-operations for this unit) involved 21 locators, including 2 fixed locators. (It is interesting to note that relatively large errors will occasionally occur even with a system " in control.") Using only the fixed locators, that unit never got lost and never had an error greater than 18m . This anomalous behavior suggests that perhaps one should use only the ranges from fixed units, if there are enough, and should use at most a small number of ranges from moving units in any case. Also worthy of note is the fact that apart from this unit, every other maximum error greater than 31m occurred with 4 or fewer locators.

As mentioned earlier, it became obvious very quickly that low-flying aircraft could not be tracked from the ground. To study the effects on accuracy of such variables as number of locators and height, a separate study was done on aircraft. 20 reporters were randomly positioned on a 10×10 km piece of terrain, and kept fixed. (See Fig. 1.) Ten (simulated) aircraft were started at the southwest corner, at altitudes of 100, 250, 400, 550, 700, 1000, 1500, 2000, 2500, and 3500m. Each flew a level course, with constant x-velocity; the course reflected several times off both north and south boundaries, terminating at the eastern boundary, systematically covering the entire terrain. (Shown on Fig. 1.) Each aircraft went through either 1230 or 2790 location-operations. The average error and average squared error were tabulated, both in the x,y-directions and in z, by number of locators, for each aircraft (i.e., each height). These tabulations were then studied, and regression techniques applied.

The first run counted locators for which the range was less than $8 \times \text{height}$, $12 \times \text{height}$, $16 \times \text{height}$, etc., up to $80 \times \text{height}$, stopping as soon as 4 or more locators were found. It then used LSS (3d). Table 2 shows the numbers of trials for which the algorithm was and was not successful by aircraft height and number of locators. There are few cases where an aircraft at 700m or higher was not located, but things are not too good below that height. The error statistics bore this out; the maximum z-errors for the 10 aircraft were 1880, 192, 97, 83, 39, 22, 14, 15, 16, 19m (in order of increasing height), while the maximum x,y-errors were 26339, 2373, 337, 600, 72, 36, 29, 23, 27, 18m respectively. Since 3d obviously does not work with

low-flying aircraft, an examination was made of the dependence of average error on number of locators and height, for aircraft heights of 550m and above. Regression analyses showed that the average x,y-error could be represented adequately by the expression $\bar{E}_{x,y} = 16 - .9n - \frac{2500}{\text{alt.}} + .017n^2$, and average z-error by $\bar{E}_z = 31 - 3n + .08 n^2$, where n is the number of locators used. (Note that these expressions were developed for aircraft higher than about 500m.) The values given by these expressions agree with the values obtained in the experiment within an average deviation of 0.5m for x,y-error, and 2m for z-error. (The experiment values for x,y-error ranged from 3 to 10m; for z-error, from 4 to 29m .)

Finally, a similar run was done using slant range reduction, with aircraft at 100, 250, 400, and 550m altitudes, using p.r.'s, and only those reporters with elevation angles $< 20^\circ$. The average x,y-errors were 3.2, 3.3, 3.4, and 3.7m respectively; the maxima were 15, 15, 15, and 17m respectively. This suggests that one should use slant range reduction up to at least 600m, and use LSS 3-d or slant range reduction above that value depending on whether there are many good ranges at large elevation angles or many at small elevation angles, and on the quality of altitude information available at different heights. (It is understood that p.r.'s of the accuracy contemplated herein are limited in altitude, but will cover at least 0 to 600m .)

Table 1

Error summaries, Boston runs.

Unit no.	All reporting		No. of times not found	Fixed units reporting		No. of times not found
	Average	Max.		Average	Max.	
7	3.8	16.6	49	5.0	21.7	
8	5.9	39.8		7.5	33.4	105
9	8.6	62.1		7.0	39.6	77
10	3.7	12.0		5.9	20.2	
11	3.4	9.3	20	5.1	22.2	
12	4.2	16.8		4.7	14.2	
13	3.7	11.2	104	5.7	21.1	104
14	3.5	10.0		6.8	32.4	
15	3.7	18.7		4.8	18.5	
16	3.7	23.5	1	4.8	16.6	
17	5.1	27.0		5.1	25.4	
18	4.9	20.4		5.3	29.5	
19	5.0	18.0		5.1	23.3	
20	3.9	15.4		5.8	30.7	
24	3.6	24.1		4.3	25.8	
25	4.3	13.3		5.0	16.3	73
26	2.9	10.0		4.1	11.8	
27	3.4	10.1		4.6	21.9	
28	6.3	108.1	2	4.5	17.7	
29	3.4	10.9		3.4	15.0	
30	5.9	51.6		4.3	12.9	79
31	7.7	79.9	8	4.2	12.3	61
32	3.7	14.1		5.1	33.9	
33	3.3	12.2		3.6	12.5	
34	3.7	12.5		3.5	20.4	
35	3.2	11.4	1	3.4	10.5	
36	3.6	11.1	1	3.1	12.8	
37	3.6	11.6		3.3	13.4	
38	2.7	10.1		3.4	10.7	
42	3.9	12.8		3.4	8.5	102
43	3.6	10.9	67	4.5	16.8	67
44	5.7	54.8	3	4.3	12.9	57
45	4.9	23.6		4.7	12.5	69
46	3.5	13.8		3.8	11.7	67
47	9.4	101.3		4.0	11.9	55
48	4.8	27.7	3	4.4	14.2	77
49	3.9	12.8	2	4.2	13.4	2
50	8.3	88.3		5.9	26.7	30
51	5.0	57.5		4.4	19.2	20
52	3.7	14.4		5.2	17.5	
53	3.8	13.5		4.7	23.1	77
54	3.8	13.1		4.6	25.9	105
55	3.9	17.2		5.4	25.5	
56	4.2	16.4		5.7	22.0	
All	4.5	108.1		4.7	39.6	

Table. 2.

Lost and found summary aircraft study

Aircraft height

Number of Locators	100	250	400	550	700	1000	1500	2000	2500	3500
3	4/0									
4	136/640	84/396	44/175	18/80	7/52					
5	40/255	48/233	28/189	18/92	3/20					
6	18/98	18/163	8/131	9/103	0/11					
7	7/24	14/98	22/192	16/128	4/32	0/1				
8	1/5	6/57	4/122	2/119	0/94	0/2				
9	0/2	6/83	3/189	5/159	1/57	0/1				
10		1/22	2/95	1/115	1/70	0/3				
11		0/1	0/16	0/85	1/98	0/0				
12			0/10	0/94	1/130	1/28				
13				0/90	0/44	0/13				
14				0/70	0/65	0/36				
15				0/16	2/228	0/184				
16				0/10	0/133	0/154				
17					0/44	0/212				
18					0/33	0/231				
19						0/232	0/67			
20						0/132	1/1162	2/1228	0/1230	0/1230
Total Lost	206	177	111	69	19	1	1	2	0	0

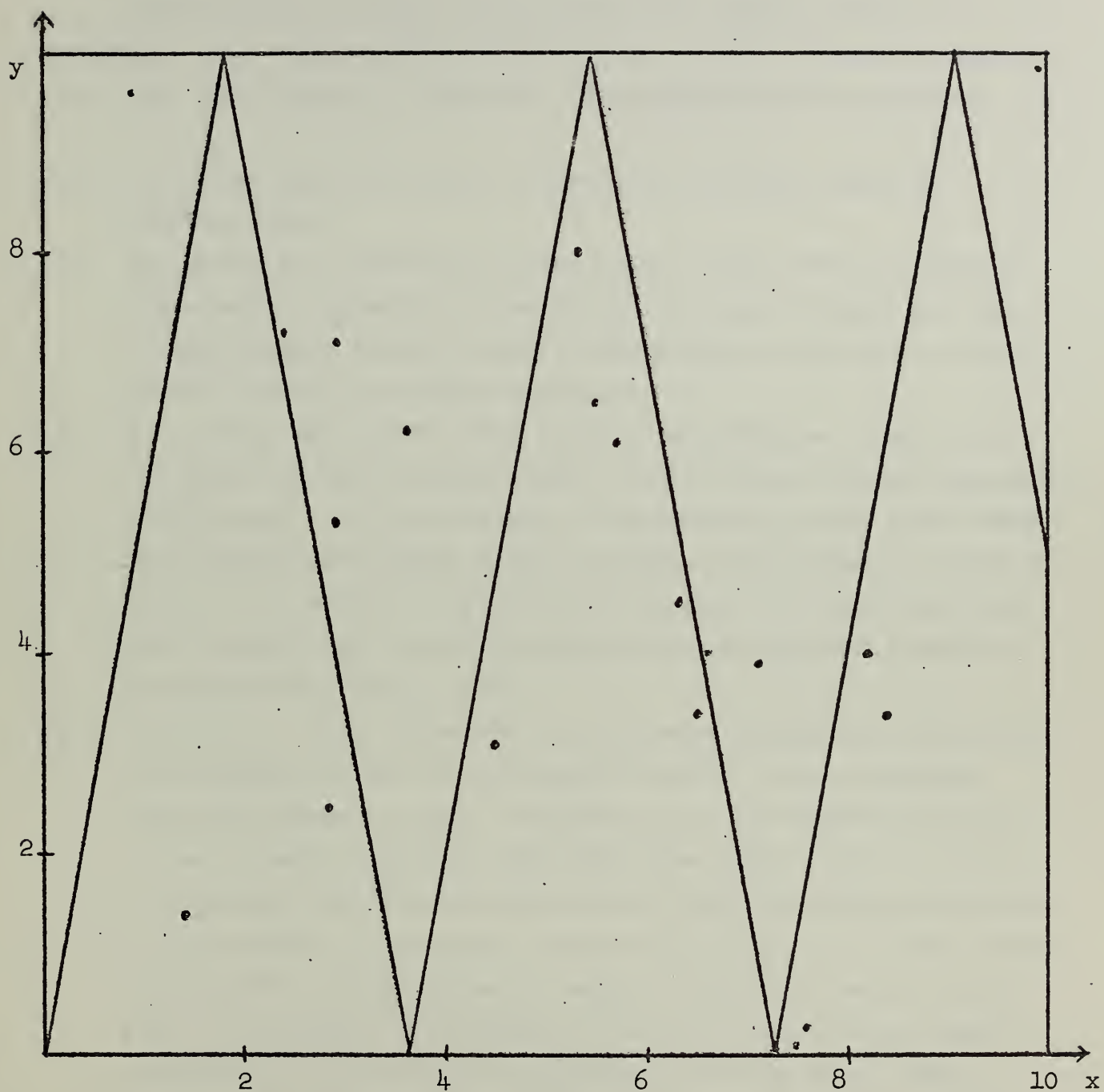


Fig. 1

Locators' positions and aircraft path, Aircraft error study.

4. Conclusions

The following represent conclusions with varying degrees of certainty; some only say what might be true, or what should be checked out. They are organized to parallel the earlier list of questions.

- (1) The system will work under reasonable conditions, such as outlined below.
- (2) The system will generally locate ground units within reasonable error bounds whenever such units have at least 4 fixed locators or well-located moving locators (which may be airplanes, as long as they are not too nearly overhead).
- (3) x, y-errors will be the same order of magnitude as range errors, for about 4-6 good locators; the accuracy (error) should improve as $1/\sqrt{n-2}$ where n is the number of locators, up to the point where the computer word length begins to limit the algorithm's accuracy. The z-errors will be comparable for genuine 3-d situations; they will depend solely on the accuracy of the independent pressure reference when that is used.
- (4) Use LSS; to find a starting value when extrapolation from previous estimates is not to be trusted, use LSL. Use an outlier-rejection scheme in each, in addition to a preliminary screening of the incoming ranges to eliminate gross (10%) errors. Use the 2-dimensional version on ground units, with slant-range reduction if the terrain is not smooth (unless all but 1 or 2 of the locators are aircraft at high elevation angles, say 60° or more, in which case 3-d treatment is required); also use 2-d with slant-range reduction for aircraft with altitudes less than about 1000m, possibly in conjunction with 3d for aircraft between (say) 600 and 1000m. Use 3d for aircraft over 1000m. (Note: These numbers depend on the spread of ground units; the values given are appropriate for the Boston deployment.)

- (5) Fixed units should constitute a large fraction (at least $1/2$) of the set of reporters. The system will work with fewer, but the possibility exists that the set of users' estimated positions will "break its moorings" and wander away, as one entity, from the true position.
- (6) It is not necessary that the same units be fixed throughout the period of time that the system is in use. (In which case it follows that the positions in which they stop will not be "known", only estimated.)
- (7) This question, concerning the number of reporters needed, is answered under (3) and (4) above.
- (8) It is understood that maps will not be generally available, but this is fortunate in a way, since their use extracts a very heavy penalty in computation time. Altimeters (or more accurately, "pressure references", or p.r.'s) will be necessary when aircraft at low elevation angles are working with ground units, either as locatee or as locators, and will be helpful whenever the terrain is rough enough that altitude differences among ground units are as large as the error σ of the p.r.
- (9) An accuracy of 14m (σ) in p.r. values (that is, $\sigma = 20\text{m}$ for the difference between the heights of two units) will enable the system to maintain position errors comparable to range-measurement errors. Better p.r.'s will of course permit more accurate position estimates.
- (10) Reporting once per 30sec. seems reasonable, except: a) Moving locators should report more frequently, especially aircraft; and b) if the system is to be used for aircraft guidance, close support, or fire control, then all aircraft should report more often, say once every second or two.
- (11) To use genuine 3d, a minimum of 3 locators is required; only one need be an aircraft. However, at least 5 (including 2 aircraft) is advisable, so that bad measurements may be detected. Note that aircraft at low elevation angles count as ground units here.
- (12) and (13) - See answer to (4).

- (14) Use as many locators as possible, but properly weighted.
- (15) Weight fixed, well-known locators considerably higher than others.
(See discussion of weights earlier in this section.)
- (16) Continual relocation is not advisable. Periodic relocation may be worthwhile; compare with (19) below.
- (17) Do not use only fixed units as reporters, except in situations with unusually good intervisibility. (There are too many times when a unit won't have enough fixed locators.)
- (18) There are indications that relatively large errors will occur when a locator begins to move, if the system continues to treat his position as fixed. Also, whatever time it takes to establish that a unit has stopped becomes a delay in "fixing" that unit's position so that he can become a good locator. Therefore a definite penalty is attached to not having a moving/stopped indicator. The size of this penalty can only be established after the appropriate filtering techniques are implemented. Thus a complete answer awaits a development study.
- (19) Indications are that it is possible to resurvey the set of fixed units. In the course of system development, attention should be paid to developing, implementing, and testing a multivariable least-squares approach, allowing the adjustment of all but a few of the fixed reporters' position estimates simultaneously. This technique would be applied periodically during long battles, when the set of currently-fixed reporters becomes considerably different from the set of initially-fixed reporters. (This technique might be either a supplement to, or a replacement for, the periodic use of relocation; see (16) above.)

Appendix

ERROR PROPAGATION FOR ONE-DIMENSIONAL LOCATION OF TWO UNITS

M.H. Pearl

March , 1972

ABSTRACT

The analysis of error propagation for position location in a one-dimensional geometry, assuming Gaussian error distributions and maximum-likelihood estimation (i.e., location by minimizing a particular quadratic penalty function), is mathematically tractable. Although this scenario is unrealistic, "solving" it in closed form may yield insights applicable to more general situations. Working Paper No. 4 developed formulas for the law of error propagation through a single location-operation in such a situation. The present note proceeds to treat a series of location-operations for the (simplest) case where only two units are involved, and shows that in this case the precision of location stabilizes over time. It is planned next to seek to extend this encouraging result to more than two units.

Note: Project working papers are informal documents prepared to facilitate discussion and communication; they may contain tentative or relatively unchecked material.

0. INTRODUCTION

Following the terminology of other papers in this series, we shall use the term location-operation to refer to the process of imputing a position to one unit (the locatee), on the basis of its measured distances from a set of other units (the observers) whose estimated positions are at hand. Both the distance measurements, and the estimates of the observers' locations, are subject to error. The purpose of Project "WHERE" can be roughly described, from a technical viewpoint, as that of determining (for a variety of scenarios) the propagation of location-error over a series of such operations, in which the role of "locatee" varies cyclically among a large set of units, many of which may be in motion during part or all of the series. As a principal tool for this purpose, project staff are developing the various components of a computerized simulation ("WHERE SIM").

While such a simulation is needed for the examination of error-propagation in "general" or "unpatterned" situations, it is plausible that valuable insights can be obtained from the study of special situations which are simple enough to admit direct mathematical analysis. Working Paper No. 4⁽¹⁾ initiated one such study, whose key simplifying assumptions are those of

- (a) one-dimensional geometry, and
- (b) Gaussian error distributions.

Specifically, WP4 derived the error-propagation formula under those assumptions for a single location-operation, leaving for a next phase of work the recursive application of this formula to determine error build-up through a sequence of location-operations, with various units serving successively as locatee. The present paper carries out this further analysis for an especially simple situation, that in which only two units are involved. The units thus alternate between the roles of locatee and observer.

(1) Project "WHERE" Working Paper No. 4, Error Propagation for Position Location in One Dimension, M.H. Pearl, December, 1971. To be referred to in what follows as "WP4."

The analysis in WP4 was divided into two cases, according as the single location-operation considered there did or did not make use of a prior estimate of the locatee's position. From the results in WP4 (see its last page) and a variety of additional evidence developed in other parts of Project "WHERE" , it now seems clear that such prior estimates should indeed be employed, and so their use is assumed in what follows. Specifically, each such prior estimate is obtained from (i) the unit's estimated position at the end of the last previous location-operation and (ii) its velocity of motion (assumed known exactly) in the interim.

With the decision between "prior-using" and "prior-ignoring" thus made in favor of the former, the analysis of the present paper is split into two cases along a different line of cleavage. In Working Paper No. 2⁽²⁾, it was suggested that the measurements and calculations taking place during a location-operation might be exploited not only to locate the locatee, but also to revise the currently assumed positions of the observers, so that one would in effect have a "location-and-revision" operation. Accordingly, both "with revision" and "without revision" scenarios will be analyzed below. The most significant finding is that (in both cases) the precision of location stabilizes over time. (The natural next step is to investigate whether this encouraging result combines to hold for more than two units.)

Although familiarity with WP4 is not essential, it will be quite helpful in reading the present text, which is organized as follows. Section 1, appearing next, describes the underlying mathematical models involved and sets up most of the necessary notation. Section 2 presents the error-propagation analysis for a sequence of "location-with-revision" operations, while Section 3 does the same for the "without revision" case.

(2) Project "WHERE" Working Paper No. 2, An Iterative Approach to the Location-Operation, A.J. Goldman, October, 1971.

1. MATHEMATICAL FORMULATION

The analysis deals with 2 units located on a line (i.e., the geometry is one-dimensional). The relevant time period begins at time $t_0 = 0$, and the successive location-operations occur at times t_j ($j = 1, 2, \dots$) where

$$0 = t_0 < t_1 < t_2 < \dots$$

For $i = 1, 2$ and $j = 0, 1, 2, \dots$, we set

$$U_{ij} = \text{true position of unit } i \text{ at time } t_j,$$

so that

$$D_j = U_{2j} - U_{1j} \tag{1.1}$$

= true (signed) distance between
the units at time t .

The U_{ij} 's are unknown to the location-system, which is however "given" at time t_j quantities,

$$u_{ij} = \text{prior estimate of } U_{ij} \text{ at time } t_j,$$

to be discussed further below. The information on which the j -th location-operation is based consists of u_{1j} , u_{2j} , and

$$d_j = \text{measured value of } D_j.$$

This measurement is assumed unbiased, i.e. $E(D_j) = d_j$, and also normally distributed, so that the random measurement error

$$\delta_j = d_j - D_j \tag{1.2}$$

is Gaussian and satisfies (for some $\sigma > 0$)

$$E(\delta_j) = 0, \quad E(\delta_j^2) = \sigma^2. \quad (1.3)$$

Note that our notation involves σ and not σ_j ; i.e., we assume no systematic change over time in the quality of the distance measurements.

The initial position-estimates, u_{10} and u_{20} , are assumed to be unbiased, i.e. $E(u_{i0}) = U_{i0}$, and also to be normally distributed; in terms of the vector u_0^T defined by

$$u_0^T = (u_{10}, u_{20}),$$

we set

$$Q_0 = \text{var}(u_0) = \begin{bmatrix} \text{var}(u_{10}) & \text{cov}(u_{10}, u_{20}) \\ \text{cov}(u_{10}, u_{20}) & \text{var}(u_{20}) \end{bmatrix}; \quad (1.4)$$

the matrix Q_0 (describing the quality of our starting position-data), as well as σ (which describes the quality of distance measurements), are the basic inputs to our analysis of error propagation.

It is assumed that between t_{j-1} and t_j , unit i moves with known velocity v_{ij} ; it therefore covers a known distance

$$\Delta_{ij} = v_{ij}(t_j - t_{j-1}). \quad (1.5)$$

Thus the prior estimates of the two units' positions, just before the first location-operation, are

$$u_{i1} = u_{i0} + \Delta_{i0} \quad (i = 1, 2);$$

it follows from (1.4) that

$$Q_0 = \text{var}(u_1) , \quad (1.6)$$

where we have used for $j = 1$ the notation

$$u_j^T = (u_{1j}, u_{2j}) .$$

Let us define the vector U_j by

$$U_j^T = (U_{1j}, U_{2j}) .$$

The calculations of the j -th location-operation, in our present model, yield

$$U_j^* = \text{maximum-likelihood estimate of } U_j \\ \text{based on the data } u_j \text{ and } d_j ;$$

this estimate is known (cf. WP4) to be unbiased, and we put

$$Q_j^* = \text{var}(U_j^*) . \quad (1.7)$$

It is at this point that the "with revision" and "without revision" cases diverge. In the first case, U_j^* is accepted as the vector of location estimates after the j -th location-operation; thus Q_j^* represent the quality of our position locations at time j , so that our aim is to determine these matrices Q_j^* (and in particular, their behavior as j increases). Moreover, in this case the "next" set of prior estimates is given by

$$u_{j+1} = U_j^* + \Delta_j , \quad (1.8)$$

where we have defined Δ_j by

$$\Delta_j^T = (\Delta_{1j} , \Delta_{2j}) .$$

In the second ("without revision") case, the vector of location estimates after the j -th operation, denoted U_j^{**} with components U_{1j}^{**} and U_{2j}^{**} , is taken as

$$\begin{aligned} U_{1j}^{**} &= U_{1j}^* , \quad U_{2j}^{**} = u_{2j} && \text{if } j \text{ odd (unit 1 the locatee) ,} \\ U_{1j}^{**} &= u_{1j} , \quad U_{2j}^{**} = U_{2j}^* && \text{if } j \text{ even (unit 2 the locatee) ,} \end{aligned} \quad (1.9)$$

so that the prior estimate of the observer's position indeed goes unrevised. The "updating" formula (1.8) is of course replaced by

$$u_{j+1} = U_j^{**} + \Delta_j . \quad (1.10)$$

And now the matrices

$$Q_j^{**} = \text{var}(U_j^{**}) \quad (1.11)$$

are the matrices which are to be investigated, and whose limiting behavior (for large j) is to be investigated.

The analyses of the two sequences $\{Q_j^*\}$ and $\{Q_j^{**}\}$ of matrices, are the topics of Section 2 and 3 respectively.

2. ANALYSIS ASSUMING REVISION

It is convenient to define the random error vectors

$$\epsilon_j = u_j - U_j \quad . \quad (2.1)$$

Then the likelihood function for the first location-operation is

$$L = K \exp\{-\frac{1}{2}(\epsilon_1^T Q_0^{-1} \epsilon_1 + \sigma^{-2} \delta_1^2)\}$$

where K is a positive constant, so that maximizing L is equivalent to minimizing

$$L^* = \frac{1}{2}(\epsilon_1^T Q_0^{-1} \epsilon_1 + \sigma^{-2} \delta_1^2) \quad . \quad (2.2)$$

From (1.1) and (1.2) ,

$$\delta_1 = d_1 + [1, -1]U_1 \quad ,$$

and from this, (2.1), and (2.2), it follows that the maximum-likelihood estimate U_1^* is the value of U_1 which minimizes

$$\begin{aligned} L^* = \frac{1}{2}\{ & (u_1 - U_1)^T Q_0^{-1} (u_1 - U_1) \\ & + \sigma^{-2} (d_1 + [1, -1] U_1)^T (d_1 + [1, -1] U_1) \} \quad . \end{aligned} \quad (2.2a)$$

We next apply the (straightforwardly verified) formula

$$\partial[(AX + B)^T S(AX + B)]/\partial X = 2A^T S(AX + B) \quad (2.3)$$

where X is a column vector and S is a symmetric matrix. Taking $A = I$ (the 2×2 identity matrix), $B = u_1$ and $S = Q_0^{-1}$, we obtain

$$\partial\{u_1 - u_1\}^T Q_0^{-1}(u_1 - u_1) / \partial u_1 = 2Q_0^{-1}(u_1 - u_1) ; \quad (2.4)$$

taking $A = [1, -1]$, $B = d_1$, and $S = I$, we obtain

$$\begin{aligned} \partial\{(d_1 + [1, -1]u_1)^T (d_1 + [1, -1]u_1)\} / \partial u_1 &= 2[1, -1]^T (d_1 + [1, -1]u_1) \\ &= 2\{[1, -1]^T d_1 + Nu_1\} , \end{aligned} \quad (2.5)$$

where we have put

$$N = [1, -1]^T [1, -1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} . \quad (2.6)$$

Thus the equation $\partial L^* / \partial u_1 = 0$, to be satisfied by $u_1 = u_1^*$, reads

$$Q_0^{-1}(u_1 - u_1) + \sigma^{-2}[1, -1] d_1 + \sigma^{-2} Nu_1 = 0 ;$$

setting

$$H_0 = \sigma^{-2} Q_0 N , \quad (2.7)$$

we see that this is equivalent to

$$(I + H_0)u_1^* = u_1 + \sigma^{-2} Q_0 [1, -1] d_1 ,$$

so that

$$u_1^* = (I + H_0)^{-1} (u_1 + \sigma^{-2} Q_0 [1, -1] d_1) . \quad (2.8)$$

Since $\text{cov}(u_1, d_1) = 0$, we have by (1.6) and (1.8)

$$\begin{aligned} \text{var}(u_1 + \sigma^{-2} Q_0 [1, -1] d_1) \\ = Q_0 + \sigma^{-4} Q_0 [1, -1] \sigma^2 [1, -1]^T Q_0 \end{aligned}$$

$$\partial\{u_1 - u_1\}^T Q_0^{-1}(u_1 - u_1)\} / \partial u_1 = 2Q_0^{-1}(u_1 - u_1) ; \quad (2.4)$$

taking $A = [1, -1]$, $B = d_1$, and $S = I$, we obtain

$$\begin{aligned} \partial\{(d_1 + [1, -1]u_1)^T (d_1 + [1, -1]u_1)\} / \partial u_1 &= 2[1, -1]^T (d_1 + [1, -1]u_1) \\ &= 2\{[1, -1]^T d_1 + N u_1\} , \end{aligned} \quad (2.5)$$

where we have put

$$N = [1, -1]^T [1, -1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} . \quad (2.6)$$

Thus the equation $\partial L^* / \partial u_1 = 0$, to be satisfied by $u_1 = u_1^*$, reads

$$Q_0^{-1}(u_1 - u_1) + \sigma^{-2}[1, -1] d_1 + \sigma^{-2} N u_1 = 0 ;$$

setting

$$H_0 = \sigma^{-2} Q_0 N , \quad (2.7)$$

we see that this is equivalent to

$$(I + H_0)u_1^* = u_1 + \sigma^{-2} Q_0 [1, -1] d_1 ,$$

so that

$$u_1^* = (I + H_0)^{-1} (u_1 + \sigma^{-2} Q_0 [1, -1] d_1) . \quad (2.8)$$

Since $\text{cov}(u_1, d_1) = 0$, we have by (1.6) and (1.8)

$$\begin{aligned} \text{var}(u_1 + \sigma^{-2} Q_0 [1, -1] d_1) \\ = Q_0 + \sigma^{-4} Q_0 [1, -1] \sigma^2 [1, -1]^T Q_0 \end{aligned}$$

$$= Q_o + \sigma^{-2} Q_o N Q_o = (I + H_o) Q_o . \quad (2.9)$$

Since (2.8) yields

$$\begin{aligned} Q_1^* &= \text{var}(U_1^*) \\ &= (I + H_o)^{-1} \text{var}(u_1 + \sigma^{-2} Q_o [1, -1] d_1) (I + H_o)^{-T} , \end{aligned}$$

it follows from (2.9) that

$$Q_1^* = Q_o (I + H_o)^{-T} , \quad (2.10)$$

which is equivalent to

$$Q_1^* = (I + H_o)^{-1} Q_o . \quad (2.11)$$

We turn now to the second location-operation. By (1.8),

$$\text{var}(u_2) = \text{var}(U_1^*) = Q_1^* ,$$

i.e. Q_1^* plays the same role now that was played by Q_o previously. Thus, if by analogy with (2.7) we put

$$H_1^* = \sigma^{-2} Q_1^* N ,$$

then by analogy with (2.10) and (2.11) we will obtain

$$Q_2^* = Q_1^* (I + H_1^*)^{-T} = (I + H_1^*)^{-1} Q_1^* .$$

More generally, the transition from Q_j^* to Q_{j+1}^{**} is given by the two equations

$$H_j^* = \sigma^{-2} Q_j^* N \quad (2.12)$$

$$Q_{j+1}^* = Q_j^* (I + H_j^*)^{-T} = (I + H_j^*)^{-1} Q_j^* , \quad (2.13)$$

with initial conditions $Q_0^* = Q_0$ and $H_0^* = H_0$.

Using these results, we will prove by induction on j that

$$Q_j^* = (I + jH_0)^{-1} Q_0 = Q_0 (I + jH_0)^{-T} . \quad (2.14)$$

By (2.11), it is true for $j = 1$. Suppose it is true for some particular value of j ; we will prove it correct for $j + 1$. First, by (2.12) and (2.14),

$$H_j^* = \sigma^{-2} (I + jH_0)^{-1} Q_0 N = (I + jH_0)^{-1} H_0 = H_0 (I + jH_0)^{-1} . \quad (2.15)$$

If we set

$$Z_j = (I + H_j^*)^{-1} ,$$

then

$$\begin{aligned} I &= Z_j (I + H_j^*) , \\ &= Z_j + Z_j H_0 (I + jH_0)^{-1} , \end{aligned}$$

so that

$$(I + jH_0) = Z_j (I + jH_0) + Z_j H_0 = Z_j [I + (j+1)H_0]$$

and thus

$$Z_j = (I + jH_0) [I + (j+1)H_0]^{-1} = [I + (j+1)H_0]^{-1} (I + jH_0);$$

it follows from (2.13 - 15) that

$$\begin{aligned}
 Q_{j+1}^* &= Z_j Q_j^* \\
 &= [I + (j+1)H_0]^{-1} [I + jH_0] [I + jH_0]^{-1} Q_0 \\
 &= [I + (j+1)H_0]^{-1} Q_0,
 \end{aligned}$$

so that (2.14) also applies for $j+1$. The induction proof is complete.

Note that with (2.6), (2.7) and (2.14), we have succeeded in determining the matrices Q_j^* explicitly in terms of the basic data Q_0 and σ . We now prove that the limit

$$Q_\infty^* = \lim_{j \rightarrow \infty} Q_j^* \quad (2.16)$$

exists, i.e. that the precision of position-location stabilizes over time, and moreover we shall evaluate Q_∞^* explicitly.

Let

$$Q_0 = \begin{bmatrix} q_1 & q \\ q & q_2 \end{bmatrix};$$

then by (2.6 - 7)

$$\begin{aligned}
 I + jH_0 &= \begin{bmatrix} 1 + \sigma^{-2}j(q_1 - q) & \sigma^{-2}j(q - q_1) \\ \sigma^{-2}j(q - q_2) & 1 + \sigma^{-2}j(q_2 - q) \end{bmatrix}, \\
 &= \sigma^{-2}j \begin{bmatrix} (\sigma^2/j) + (q_1 - q) & q - q_1 \\ (q - q_2) & (\sigma^2/j) + (q_2 - q) \end{bmatrix}.
 \end{aligned} \quad (2.17)$$

The coefficient of $\sigma^{-2}j$ is a matrix with determinant

$$(\sigma^4/j^2) + (\sigma^2/j)(q_1 + q_2 - 2q) = (\sigma^2/j)[(\sigma^2/j) + (q_1 + q_2 - 2q)] ,$$

and thus with inverse matrix

$$(\sigma^2/j)^{-1} [(\sigma^2/j) + (q_1 + q_2 - 2q)]^{-1} \begin{bmatrix} (\sigma^2/j) + (q_2 - q) & q_1 - q \\ q_2 - q & (\sigma^2/j) + (q_1 - q) \end{bmatrix} .$$

It follows from (2.17) that

$$(I + jH_0)^{-1} = [(\sigma^2/j) + (q_1 + q_2 - 2q)]^{-1} \begin{bmatrix} (\sigma^2/j) + (q_2 - q) & q_1 - q \\ q_2 - q & (\sigma^2/j) + (q_1 - q) \end{bmatrix} . \quad (2.18)$$

In particular,

$$\lim_{j \rightarrow \infty} (I + jH_0)^{-1} = H^* \quad (2.19)$$

exists and is given by

$$H^* = (q_1 + q_2 - 2q)^{-1} \begin{bmatrix} q_2 - q & q_1 - q \\ q_2 - q & q_1 - q \end{bmatrix} ; \quad (2.20)$$

thus by (2.14) the limit Q_{∞}^* exists and is given by

$$\begin{aligned} Q_{\infty}^* &= H^* Q_0 \\ &= (q_1 + q_2 - 2q)^{-1} (q_1 q_2 - q^2) J, \end{aligned} \quad (2.21)$$

where J is a 2×2 matrix of ones. It is somewhat surprising that the limit is independent of σ .

A natural question is whether the position-locations grow increasingly precise over time, i.e. whether the diagonal entries of $Q_j^* = \text{var}(U_j^*)$ are decreasing functions of j . By (2.14) and (2.18), the (1,1) entry of Q_j^* is

$$\text{var}(U_{1j}^*) = q_1 [(\sigma^2/j) + q_1^{-1}(q_1 q_2 - q^2)] / [(\sigma^2/j) + (q_1 + q_2 - 2q)] \quad (2.22)$$

Now it is readily verified that the function

$$f(x) = (x+a)/(x+b) = 1 - (b-a)/(x+b)$$

is increasing or decreasing according as $(b-a)$ is positive or negative.

Applying this with

$$x = \sigma^2/j, \quad a = q_1^{-1}(q_1 q_2 - q^2), \quad b = q_1 + q_2 - 2q,$$

we see that $\text{var}(U_{1j}^*)$ is a non-increasing function of j if

$$q_1 + q_2 - 2q \geq q_1^{-1}(q_1 q_2 - q^2), \quad (2.23)$$

which indeed holds since it is equivalent to $(q_1 - q)^2 \geq 0$. Analogously, $\text{var}(U_{2j}^*)$ is a non-increasing function of j .

3. ANALYSIS FOR LOCATION WITHOUT REVISION

We now turn to the matrices

$$Q_j^{**} = \text{var}(U_j^{**}) \quad (3.1)$$

corresponding to the location-without-revision scenario. The analysis is more complicated here, since the two units no longer figure symmetrically in each location-operation (recall that unit 1 is the locatee in odd-numbered operations, unit 2 the locatee in even-numbered ones).

It is convenient to repeat from Section 2 the definitions

$$N = [1, -1]^T [1, -1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (3.3)$$

$$H_o = \sigma^{-2} Q_o N, \quad (3.3)$$

and to introduce the matrices E_{ik} ; E_{ik} has entry 1 in the (row i , column k) position, and all other entries zero. Thus, for example, the consequence

$$(U_1^{**})^T = [U_{11}^*, 0]^T + [0, u_{21}]^T$$

of (1.9) can be written

$$U_1^{**} = E_{11} U_1^* + E_{22} u_1. \quad (3.4)$$

From this and (3.1) it follows that

$$\begin{aligned} Q_1^{**} &= E_{11} \text{var}(U_1^*) E_{11} + E_{22} \text{var}(u_1) E_{22} \\ &+ E_{11} \text{cov}(U_1^*, u_1) E_{22} + E_{22} \text{cov}(u_1, U_1^*) E_{11} . \end{aligned} \quad (3.5)$$

By (2.8) and (2.11)

$$\text{cov}(U_1^*, u_1) = (I + H_0)^{-1} Q_0 = Q_1^* ,$$

so that (3.5) yields

$$\begin{aligned} Q_1^{**} &= E_{11} Q_1^* E_{11} + E_{22} Q_0 E_{22} \\ &+ E_{11} Q_1^* E_{22} + E_{22} Q_1^* E_{11} \\ &= Q_1^* + E_{22} (Q_0 - Q_1^*) E_{22} . \end{aligned}$$

By use of (2.11), this reads

$$Q_1^{**} = (I + H_0)^{-1} Q_0 + E_{22} [I - (I + H_0)^{-1}] Q_0 E_{22} . \quad (3.6)$$

The second location-operation follows the same logic as the first, except that Q_1^{**} plays the part of Q_0 and that the roles of the two units are interchanged. Thus, by analogy with (3.3) and (3.6), we have

$$H_1^{**} = \sigma^{-2} Q_1^{**} N , \quad (3.7)$$

$$Q_2^{**} = (I + H_1^{**})^{-1} Q_1^{**} + E_{11} [I - (I + H_1^{**})^{-1}] Q_1^{**} E_{11} . \quad (3.8)$$

More generally, the appropriate recursion equations are

$$H_j^{**} = \sigma^{-2} Q_j^{**} N \quad (3.9)$$

$$Q_{2j+1}^{**} = (I + H_{2j}^{**})^{-1} Q_{2j}^{**} + E_{22} [I - (I + H_{2j}^{**})^{-1}] Q_{2j}^{**} E_{22} , \quad (3.10)$$

$$Q_{2j+2}^{**} = (I + H_{2j+1}^{**})^{-1} Q_{2j+1}^{**} + E_{11} [I - (I + H_{2j+1}^{**})^{-1}] Q_{2j+1}^{**} E_{11}; \quad (3.11)$$

these are accompanied by the initial conditions $Q_0^{**} = Q_0$ and $H_0^{**} = H_0$.

In view of the way in which the analysis "alternates" between the two units, it would be unreasonable to expect the type of convergence given by the existence of a single limiting matrix

$$Q_\infty^{**} = \lim_{j \rightarrow \infty} Q_j^{**} .$$

Rather, one might hope for "convergence with period 2", i.e. for the existence of a pair of matrices

$$Q_\infty^o = \lim_{j \rightarrow \infty} Q_{2j+1}^{**} , \quad Q_\infty^e = \lim_{j \rightarrow \infty} Q_{2j}^{**} , \quad (3.12)$$

where the superscripts "o" and "e" stand for "odd" and "even" respectively. Furthermore, in view of the results obtained in Section 2, one would not be surprised if these two limiting matrices turned out to be independent of σ .

At present we have not succeeded in analyzing this situation with anything like the completeness achieved in Section 2 for the "location with revision" case. However, we have succeeded in proving that in the present

case, like that of Section 2, the precision of position-location stabilizes over time. To make this precise, let

$$Q_j^{**} = \begin{bmatrix} q_1^{(j)} & q^{(j)} \\ q^{(j)} & q_2^{(j)} \end{bmatrix} ;$$

then $q^{(0)} = q$ and $q_i^{(0)} = q_i$ ($i = 1, 2$). The precision with which the j -th location-operation estimates the position of the locatee is given by

$$\begin{aligned} q_2^{(j)} &= \text{var}(U_{2j}^*) & \text{if } j \text{ is even,} \\ q_1^{(j)} &= \text{var}(U_{1j}^*) & \text{if } j \text{ is odd.} \end{aligned}$$

It will be shown below that

$$\text{sequences } \{q_2^{(2j)}\} \text{ and } \{q_1^{(2j+1)}\} \text{ are convergent,} \quad (3.13)$$

thus proving the "stabilization" mentioned above.

The logic of the proof will run as follows. It will be shown that

$$\text{sequences } \{q_1^{(2j)}\} \text{ and } \{q_2^{(2j)}\} \text{ are convergent.} \quad (3.14)$$

This is a property of the sequence $\{Q_{2j}^*\}$ of even-index matrices. Since the sequence $\{Q_{2j+1}^*\}$ of odd-index matrices obeys the same type of recursion, and since (3.14) is symmetric as between the two units, it follows by analogy

that

$$\text{sequences } \{q_1^{(2j+1)}\} \text{ and } \{q_2^{(2j+1)}\} \text{ are convergent.} \quad (3.15)$$

The desired result (3.13) then follows from (3.14) and (3.15).

Thus it suffices to prove (3.14). Since the variance terms $q_1^{(2j)}$ and $q_2^{(2j)}$ are nonnegative, i.e. bounded below by 0, it suffices to prove that the sequences in (3.14) are non-increasing. These statements,

$$q_1^{(2j+2)} \leq q_1^{(2j)} \quad , \quad q_2^{(2j+2)} \leq q_2^{(2j)} \quad ,$$

are to be proved by induction on j , but it suffices to establish the prototype induction step, namely

$$q_1^{(2)} \leq q_1 \quad , \quad q_2^{(2)} \leq q_2 \quad . \quad (3.16)$$

The proof of the desired result (3.13) has now been reduced to the demonstration of the two statements (3.16). For proving them, it is convenient to introduce the auxiliary quantities

$$S = (q_1 - q) + (q_2 - q) \quad , \quad \Delta = q_1 q_2 - q^2 \quad . \quad (3.17)$$

Since Q_0 is a variance-covariance matrix, its determinant $\Delta \geq 0$, and it follows by (2.23) that $S \geq 0$. The following calculations will make use of the easily-checked identities

$$\Delta - qS = (q_1 - q)(q_2 - q) \quad , \quad (3.18)$$

$$\Delta - q_1 S = -(q_1 - q)^2 \quad . \quad (3.19)$$

Direct calculation from (3.6) yields

$$Q_1^{**} = (\sigma^2 + S)^{-1} \begin{bmatrix} \sigma^2 q_1 + \Delta & \sigma^2 q + \Delta \\ \sigma^2 q + \Delta & (\sigma^2 + S)q_2 \end{bmatrix} . \quad (3.20)$$

From (3.7), we obtain

$$(I + H_1^{**})^{-1} = [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \begin{bmatrix} \sigma^4 + (S + q_2 - q)\sigma^2 + (q_2 - q)^2 & \sigma^2(q_1 - q) \\ \sigma^2(q_2 - q) + (q_2 - q)^2 & \sigma^4 + (S + q_1 - q)\sigma^2 \end{bmatrix} .$$

Application of (3.8) and (3.20) yields

$$q_1^{(2)} = q_1^{(1)} = (\sigma^2 + S)^{-1}(\sigma^2 q_1 + \Delta) ,$$

so that the first of the desired statements (3.16) is equivalent to

$$\sigma^2 q_1 + \Delta \leq (\sigma^2 + S)q_1$$

and hence to $\Delta - q_1 S \leq 0$, which is true by (3.19). Next, substitution into (3.8) yields

$$q_2^{(2)} = (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \{q_2 \sigma^6 + [\Delta + 2q_2 S] \sigma^4 \\ + [2q_2 S^2 - (q_2 - q)^2 (q_2 - 2q)] \sigma^2 + \Delta (q_2 - q)^2\} ,$$

so that the second of the desired statements (3.16) is equivalent to

$$q_2 \sigma^6 + [\Delta + 2q_2 S] \sigma^4 + [2q_2 S^2 - (q_2 - q)^2 (q_2 - 2q)] \sigma^2 + \Delta (q_2 - q)^2 \\ \leq q_2 (\sigma^2 + S) [\sigma^4 + 2S \sigma^2 + (q_2 - q)^2] ,$$

which is true since it is in turn equivalent to

$$(q_2 - q)^2 [\sigma^2 + (q_2 - q)]^2 \geq 0 .$$

This completes the proof of (3.13), so that the existence of the "variance terms" in (3.12), i.e. entries q_1^0 and q_2^0 in Q_∞^0 and entries q_1^e and q_2^e in Q_∞^e , is established. We have not as yet succeeded in proving the existence of the covariance limits

$$q^0 = \lim_{j \rightarrow \infty} q^{(2j+1)} , \quad q^e = \lim_{j \rightarrow \infty} q^{(2j)} , \quad (3.21)$$

i.e. the off-diagonal terms in (3.12), except under the additional hypotheses

$$q_1 \geq q , \quad q_2 \geq q . \quad (3.22)$$

This assumption is probably not very restrictive for the situations of interest here; it asserts that the errors in the initial estimates (of the two units' positions), if positively correlated, have covariance less than either of their individual variances. Because $\Delta \geq 0$, (3.22) will automatically be satisfied if $q_1 = q_2$.

To prove that (3.22) implies (3.21), we first prove that it implies the generalization

$$q_1^{(j)} \geq q^{(j)} , \quad q_2^{(j)} \geq q^{(j)} . \quad (3.23)$$

of itself. The proof is by induction on j , and since (3.23) is symmetric as between the two units, it suffices to treat a prototype induction step, i.e. to show that

$$q_1^{(1)} \geq q^{(1)}, \quad q_2^{(1)} \geq q^{(1)} \quad (3.24)$$

follows from (3.22). From (3.20); we see that the first part of (3.24) is an immediate consequence of the first part of (3.22). Also from (3.20), we see that the second part of (3.24) is equivalent to

$$(\sigma^2 + S)q_2 \geq \sigma^2 q + \Delta, \quad ,$$

hence to

$$\sigma^2(q_2 - q) \geq \Delta - q_2 S - (q_2 - q)^2, \quad ,$$

which follows from the second part of (3.22).

We will show below that (3.22) implies

$$\text{sequence } \{q^{(2j)}\} \text{ is non-decreasing.} \quad (3.25)$$

Because the variance-covariance matrix Q_{2j}^{**} must have a non-negative determinant,

$$[q^{(2j)}]^2 \leq q_1^{(2j)} q_2^{(2j)}, \quad ,$$

it follows that the sequence in (3.25) is ultimately bounded above by a quantity $[q_1^e q_2^e]^{\frac{1}{2}} + \epsilon$ where $\epsilon > 0$, and so this sequence must have a limit q^e . This pertains to the sequence $\{Q_{2j}^{**}\}$; since the sequence $\{Q_{2j+1}^{**}\}$ is generated by the same type of recursion, and since by (3.24) its initial

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

TO THE HONORABLE CHAIRMAN OF THE BOARD OF TRUSTEES
OF THE UNIVERSITY OF CHICAGO

AND TO THE HONORABLE CHAIRMAN OF THE BOARD OF TRUSTEES
OF THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
JANUARY 1950

matrix Q_1 satisfies the analog of (3.22), the same logic proves the existence of the limiting covariance q^0 .

It only remains to prove that (3.22) implies (3.25). Since (3.22) implies (3.23), it suffices to establish the prototype induction step, namely that (3.22) implies

$$q^{(2)} \geq q. \quad (3.26)$$

Application of (3.8) yields

$$\begin{aligned} q^{(2)} &= (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \{ q\sigma^6 \\ &\quad + [(q_2 - q)q_1 + \Delta + q(S + q_1 - q)]\sigma^4 + [(q_2 - q)\Delta + q_1(q_2 - q)^2 + (S + q_1 - q)\Delta]\sigma^2 \\ &\quad + (q_2 - q)^2 \Delta \} \\ &= (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \{ q\sigma^6 \\ &\quad + [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2 - q)^2]\sigma^2 + (q_2 - q)^2 \Delta \}. \end{aligned}$$

Thus (3.26) is equivalent to

$$\begin{aligned} q\sigma^6 + [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2 - q)^2]\sigma^2 + (q_2 - q)^2 \Delta \\ \geq q(\sigma^2 + S)[\sigma^4 + 2S\sigma^2 + (q_2 - q)^2], \end{aligned}$$

or in turn to

$$(\Delta - qS)\{2\sigma^4 + [2S + (q_2 - q)]\sigma^2 + (q_2 - q)^2\} \geq 0,$$

which is true by virtue of (3.18) and (3.22). This completes the proof that the limiting covariances (3.21) exist.

matrix Q_1 satisfies the analog of (3.22), the same logic proves the existence of the limiting covariance q^0 .

It only remains to prove that (3.22) implies (3.25). Since (3.22) implies (3.23), it suffices to establish the prototype induction step, namely that (3.22) implies

$$q^{(2)} \geq q. \quad (3.26)$$

Application of (3.8) yields

$$\begin{aligned} q^{(2)} &= (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \{ q\sigma^6 \\ &\quad + [(q_2 - q)q_1 + \Delta + q(S + q_1 - q)]\sigma^4 + [(q_2 - q)\Delta + q_1(q_2 - q)^2 + (S + q_1 - q)\Delta]\sigma^2 \\ &\quad + (q_2 - q)^2 \Delta \} \\ &= (\sigma^2 + S)^{-1} [\sigma^4 + 2S\sigma^2 + (q_2 - q)^2]^{-1} \{ q\sigma^6 \\ &\quad + [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2 - q)^2]\sigma^2 + (q_2 - q)^2 \Delta \}. \end{aligned}$$

Thus (3.26) is equivalent to

$$\begin{aligned} q\sigma^6 + [2\Delta + qS]\sigma^4 + [2S\Delta + q_1(q_2 - q)^2]\sigma^2 + (q_2 - q)^2 \Delta \\ \geq q(\sigma^2 + S)[\sigma^4 + 2S\sigma^2 + (q_2 - q)^2], \end{aligned}$$

or in turn to

$$(\Delta - qS)\{2\sigma^4 + [2S + (q_2 - q)]\sigma^2 + (q_2 - q)^2\} \geq 0,$$

which is true by virtue of (3.18) and (3.22). This completes the proof that the limiting covariances (3.21) exist.

Although the main objectives in this Section have been achieved, we should note several items of "unfinished business" which need to be settled to obtain a thorough analysis:

(a) Determine whether the limiting covariances q^o and q^e (and hence, the limiting matrices Q_∞^o and Q_∞^e) exist in general, and not only under the extra hypotheses (3.22).

(b) Evaluate the various limits in closed form.

(c) If (b) is too ambitious, at least determine whether the limits are independent of σ .

PROJECT "WHERE" : WORKING PAPER NO. 7

Algorithms

D. J. Sookne

1. Description of the Algorithms used by WHERSM

The algorithm used to locate each unit is the least squares squared (LSS) algorithm. Both two- and three-dimensional subroutines are included in the WHERSM system. The three-dimensional program is described in detail below.

For each locator i , $i = 1, 2, \dots, n$, let (x_i, y_i, z_i) be its estimated position, r_i its reported distance to the locatee, and w_i the reciprocal of its variance, the calculation of which is described below. The w_i 's act as weights; the smaller the variance of a locator, the more accurately we know its position, and the higher its weight will be. Then LSS minimizes

$$E = \sum_{i=1}^n w_i E_i \quad \text{where} \quad (1)$$

$$E_i = \frac{(d_i^2 - r_i^2)^2}{4r_i^2} ; \quad (2)$$

here d_i is the calculated distance between (x_i, y_i, z_i) and (x, y, z) , the estimated position of the locatee, so

$$d_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 . \quad (3)$$

Note that

$$E_i = \frac{(d_i^2 - r_i^2)^2}{4r_i^2} = \frac{(d_i + r_i)^2}{4r_i^2} \cdot (d_i - r_i)^2 .$$

THE HISTORY OF THE

REIGN OF

CHARLES THE FIRST

BY SAMUEL JOHNSON

IN THREE VOLUMES. VOL. II.

LONDON: Printed by J. Sturges, in Pall-mall; and by J. Knapton, in Strand, 1720.

THE HISTORY OF THE REIGN OF CHARLES THE FIRST, BY SAMUEL JOHNSON, ESQ. IN THREE VOLUMES. VOL. II. LONDON: Printed by J. Sturges, in Pall-mall; and by J. Knapton, in Strand, 1720.

When the estimated positions of locators and locatee are near their true positions, r_i is very near d_i , so the factor $(d_i + r_i)^2 / 4r_i^2$ is very near 1, and E_i behaves very much like $(d_i - r_i)^2$. Thus we expect this algorithm to produce answers nearly identical to those produced by the least squares (LS) algorithm, for which $E_i = (d_i - r_i)^2$. Indeed, computer tests indicate that there is no significant difference between the results of these two algorithms. Thus LSS is used because it is more economical, since LS requires the calculation of d_i , which in turn requires a square root to be taken (see (3)). LSS avoids the square root, since only d_i^2 is needed.

Where E is a minimum, its partial derivatives with respect to x , y , and z will be zero. Thus the program calculates

$$\begin{aligned}
 F &= \frac{\partial E}{\partial x} = \sum_{i=1}^n \frac{w_i (d_i^2 - r_i^2)}{r_i^2} (x - x_i) \\
 G &= \frac{\partial E}{\partial y} = \sum_{i=1}^n \frac{w_i (d_i^2 - r_i^2)}{r_i^2} (y - y_i) \\
 H &= \frac{\partial E}{\partial z} = \sum_{i=1}^n \frac{w_i (d_i^2 - r_i^2)}{r_i^2} (z - z_i) .
 \end{aligned} \tag{4}$$

Since d_i depends on x , y , and z (see (3)), F , G , and H depend non-linearly on these variables. Then the system (4) cannot be solved directly for x , y , and z . Instead, LSS does successive Newton-Raphson iterations ([1], pp. 201-223), beginning with a starting point $p = (x, y, z)$. The first iteration yields a new point $p' = (x + \Delta x, y + \Delta y, z + \Delta z)$; this point is used as the starting point for the next iteration, and so on. The process begins with the first order approximation

$$F_p' = F_p + \frac{\partial F}{\partial x} \Delta x + \frac{\partial F}{\partial y} \Delta y + \frac{\partial F}{\partial z} \Delta z$$

$$G_p' = G_p + \frac{\partial G}{\partial x} \Delta x + \frac{\partial G}{\partial y} \Delta y + \frac{\partial G}{\partial z} \Delta z$$

$$H_p' = H_p + \frac{\partial H}{\partial x} \Delta x + \frac{\partial H}{\partial y} \Delta y + \frac{\partial H}{\partial z} \Delta z$$

Here F_p denotes the value of F at point p , and all partial derivatives are evaluated at p . Thus this system represents the expansions of the functions F , G , and H about p . Now we want F_p' , G_p' , and H_p' all to be zero, so the problem reduces to that of finding Δx , Δy , Δz which satisfy the three simultaneous equations

$$\frac{\partial F}{\partial x} \Delta x + \frac{\partial F}{\partial y} \Delta y + \frac{\partial F}{\partial z} \Delta z + F_p = 0$$

$$\frac{\partial G}{\partial x} \Delta x + \frac{\partial G}{\partial y} \Delta y + \frac{\partial G}{\partial z} \Delta z + G_p = 0 \quad (5)$$

$$\frac{\partial H}{\partial x} \Delta x + \frac{\partial H}{\partial y} \Delta y + \frac{\partial H}{\partial z} \Delta z + H_p = 0$$

The program calculates F , G , and H (see (4)), and

$$\frac{\partial F}{\partial x} = \sum_{i=1}^n \frac{w_i (d_i^2 - r_i^2 + 2(x - x_i)^2)}{r_i^2}$$

$$\frac{\partial G}{\partial y} = \sum_{i=1}^n \frac{w_i (d_i^2 - r_i^2 + 2(y - y_i)^2)}{r_i^2}$$

$$\frac{\partial H}{\partial z} = \sum_{i=1}^n \frac{w_i (d_i^2 - r_i^2 + 2(z - z_i)^2)}{r_i^2}$$

(6)

$$\begin{aligned}
\frac{\partial G}{\partial x} &= \frac{\partial}{\partial x} \frac{\partial E}{\partial y} = \frac{\partial}{\partial y} \frac{\partial E}{\partial x} = \frac{\partial F}{\partial y} = \sum_{i=1}^n \frac{2w_i(x - x_i)(y - y_i)}{r_i^2} \\
\frac{\partial H}{\partial x} &= \frac{\partial}{\partial x} \frac{\partial E}{\partial z} = \frac{\partial}{\partial z} \frac{\partial E}{\partial x} = \frac{\partial F}{\partial z} = \sum_{i=1}^n \frac{2w_i(x - x_i)(z - z_i)}{r_i^2} \\
\frac{\partial H}{\partial y} &= \frac{\partial}{\partial y} \frac{\partial E}{\partial z} = \frac{\partial}{\partial z} \frac{\partial E}{\partial y} = \frac{\partial G}{\partial z} = \sum_{i=1}^n \frac{2w_i(y - y_i)(z - z_i)}{r_i^2} ,
\end{aligned} \tag{6}$$

then solves (5) for $\Delta x, \Delta y, \Delta z$. This determines $p' = (x + \Delta x, y + \Delta y, z + \Delta z)$, which is used as the starting point for the next iteration. For a given set of locators, the program will do up to 4 iterations, stopping if an iteration converges, namely

$$\max(|F|, |G|, |H|) \leq \sum_{i=1}^n w_i ,$$

or if an iteration diverges (which see below) . I.e. convergence occurs when F, G , and H are all small.

Computer tests indicated that for both the two- and three-dimensional versions of LSS , 2 to 4 Newton-Raphson iterations are generally sufficient for convergence, even when the starting point is 100 meters from the true position.

The LSS algorithm allows for easy elimination of outliers. These are locators whose reported distance (r_i) to the locatee contains a greater error than do the reported distances of the other locators. Let

$$\bar{E} = \left(\sum_{i=1}^n w_i E_i \right) / \sum_{i=1}^n w_i . \tag{7}$$

THE UNIVERSITY OF CHICAGO

PHYSICS DEPARTMENT

PHYSICS 311

PROFESSOR OF PHYSICS

PHYSICS 311

PHYSICS 311

PHYSICS 311

PHYSICS 311

PHYSICS 311

Now any locator i is eliminated for which

$$E_i > 4\bar{E} . \quad (8)$$

Several other constants greater than 1 were tried in place of 4, but computer tests showed that 4 was best.

The algorithm iterates on all locators until convergence occurs, up to 4 iterations (for divergence, see below). Then \bar{E} is calculated and some locators may be eliminated via (8). (If no locators are eliminated, we're through). Then more iterations are performed on the reduced set of locators. After convergence, (7) and (8) are again applied to eliminate outliers. This iteration-elimination process continues until either (1) no locators are eliminated in a step, (2) 1/4 of the locators have been eliminated, or (3) the algorithm diverges (see below). In cases (1) and (2) the algorithm terminates.

There are two types of divergence; an increase in the objective function, E , from one iteration to the next and a badly scaled determinant less than 10^{-8} in absolute value which makes equations (5) impossible to solve accurately. In each case, we halve the step size of the previous iteration, and try again. This can be done unless the iteration at which divergence occurs is the first iteration for this locatee. Then we must quit without finding a solution.

For any locatee and set of locators, the size of successive iteration steps should decrease rapidly. If not, then the problem is ill-behaved, and perhaps dozens of iterations will be necessary to achieve convergence. To avoid this, the algorithm attempts to "home in" on the solution faster. If the step size in the x -direction (or y or z) is Δx_1 for one iteration and Δx_2 for the next, and if $|\Delta x_2| > \frac{1}{2} |\Delta x_1|$, then Δx_2 is replaced by

$$2\min(|\Delta x_1|, |\Delta x_2|) \cdot \text{sgn} \Delta x_2$$

if Δx_1 and Δx_2 have the same sign, and if they have opposite signs, Δx_2 is replaced by

$$\frac{\Delta x_2 \cdot \Delta x_1}{\Delta x_1 - \Delta x_2}$$

A further refinement of the algorithm was tried; this involved relocating the locators. More concerning this may be found in sec. 3 and Working Paper No. 8. Once the Newton iterations have converged, there is an error between the reported distance r_i and the calculated distance d_i , for each locator i in the set. By changing the estimated position of each locator, i.e. by moving it along the straight line connecting it with the locatee, this error $d_i - r_i$ may be reduced, even eliminated. After relocating, more Newton-Raphson iterations may be done. Several runs were made, using different values for the fraction, f , of the error eliminated by relocation, with $0 < f < 1$. The best such f was the smallest, and that was not quite as good as no relocation ($f = 0$). Relocation remains an option in the LSS routines, but one which is not now used.

The 2-dimensional LSS program is analogous to the 3-dimensional one. It solves the equations

$$\frac{\partial F}{\partial x} \Delta x + \frac{\partial F}{\partial y} \Delta y + F = 0$$

$$\frac{\partial G}{\partial x} \Delta x + \frac{\partial G}{\partial y} \Delta y + G = 0$$

Here, F and G and their partial derivatives are as described above, except that all z -components are zero. The 2-dimensional program is less likely to result in divergence of an iteration, so the special steps to handle divergence (see previous page) are omitted.

The weights w_i (see (1)) are again taken as the reciprocals of the variances σ_i^2 . When a unit has been located, the variance of its location is calculated as

$$\frac{\left(\sum_{i=1}^n w_i E_i \right)}{(n-m) \sum_{i=1}^n w_i} \quad (9)$$

when n is the number of locators, m the dimensionality (2 or 3) of the algorithm.

When the locatee is lost, so that no starting point exists for the LSS routine, the least squares linear (LSL) routine is called to provide a starting point. First, a slant-range reduction is done to convert 3-D reported distances to 2-D. Then the 2-D LSL algorithm is called. This is the same basic algorithm which is described in sec. 2 of [2]. The modifications are these: all pairs of circles are used, and outliers are eliminated. Consider the M radical axis lines each formed by a pair of circles; the equation for such a line is found by subtracting the equation of one circle from the equation of the other as in [2], p. 10. For each such line k , $k = 1, 2, \dots, M$, let d_k be the calculated distance from the locatee to the line. Then LSL minimizes

$$D = \sum_{k=1}^M d_k^2 .$$

After a locatee has been found using all M lines, the routine calculates

$$\overline{d^2} = D/M ,$$

the average squared distance to the lines. Then any line k is eliminated for which

$$d_k^2 > 8 \cdot \overline{d^2} .$$

The reduced set of locators is used to calculate a position, $\overline{d^2}$ is recalculated, and a line k is eliminated for which

$$d_k^2 > 4 \cdot \overline{d^2} .$$

The process is repeated twice more, once with the elimination condition

$$d_k^2 > 2 \cdot \overline{d^2} , \quad \text{and finally with}$$

$$d_k^2 > \overline{d^2} .$$

The LSL algorithm will fail only when the equations at the bottom of p. 13 of [2] are ill-conditioned. This will happen when the M lines are nearly parallel, i.e. the locators all lie near one straight line. In this case, no starting point can be found for LSS .

The routine FINDIT calls LSS (2-D or 3-D) and LSL when necessary. If the locatee is not lost, two previous locations are used to extrapolate in the x - and y - directions to get a starting point for LSS. The last z -value is used, with no extrapolation. Then LSS is called; the 2-dimensional routine for ground units, 3-dimensional for airplanes.

If the unit is lost, LSL is called after doing a slant-range reduction (using the last known z -value) to convert 3-dimensional distances to planar distances. Then using the planar distances, the 2-dimensional LSS is called. If the unit is an airplane, the x - and y -coordinates returned by LSS(2-D) plus the last known z -value are used as a starting point for LSS(3-D).

Control is returned to FINDIT with the variance defined by (9) . This represents the variance of the position estimate, based on the reported

distances. The total variance is computed by adding RVAR, the variance of the distances. If the locatee is known to be moving, its weight is calculated by inverting the total variance.

If the unit is fixed, the current position estimate is taken to be a weighted average of the old estimate and the just-calculated estimate. Then the weight of the locatee is computed as a weighted average of the old weight and the just-calculated weight. A further description of these calculations may be found in Section three of this report.

2. Description of Algorithms Programmed for WHERSM But Not Now Used

The linear method (LM) and smallest tangent circle (STC) method are described in [2]. They were both found to be less accurate than the LSS and LSL algorithms.

The least squares method mentioned in §1 is the same as LSS, except equations (2), (4), and (6) are replaced by

$$E_i = (d_i - r_i)^2 \quad (2')$$

$$\left. \begin{aligned} F &= \frac{1}{2} \frac{\partial E}{\partial x} = \sum_{i=1}^n \frac{(d_i - r_i)(x - x_i)}{d_i} \\ G &= \frac{1}{2} \frac{\partial E}{\partial y} = \sum_{i=1}^n \frac{(d_i - r_i)(y - y_i)}{d_i} \\ H &= \frac{1}{2} \frac{\partial E}{\partial z} = \sum_{i=1}^n \frac{(d_i - r_i)(z - z_i)}{d_i} \end{aligned} \right\} \quad (4')$$

and

$$\frac{\partial F}{\partial x} = \sum_{i=1}^n \left(\frac{d_i - r_i}{d_i} + \frac{r_i (x - x_i)^2}{d_i^3} \right)$$

$$\frac{\partial G}{\partial y} = \sum_{i=1}^n \left(\frac{d_i - r_i}{d_i} + \frac{r_i (y - y_i)^2}{d_i^3} \right)$$

$$\frac{\partial H}{\partial z} = \sum_{i=1}^n \left(\frac{d_i - r_i}{d_i} + \frac{r_i (z - z_i)^2}{d_i^3} \right)$$

(6')

$$\frac{\partial G}{\partial x} = \frac{\partial F}{\partial y} = \sum_{i=1}^n \frac{r_i (x - x_i)(y - y_i)}{d_i^3}$$

$$\frac{\partial H}{\partial x} = \frac{\partial F}{\partial z} = \sum_{i=1}^n \frac{r_i (x - x_i)(z - z_i)}{d_i^3}$$

$$\frac{\partial H}{\partial y} = \frac{\partial G}{\partial z} = \sum_{i=1}^n \frac{r_i (y - y_i)(z - z_i)}{d_i^3}$$

The other algorithm programmed was MINMAX which minimizes the maximum of the errors $d_i - r_i$. This algorithm proved to be less accurate than STC and much less accurate than LSS. Further, it uses 10 to 100 times the computer time the other algorithms use, since (in the two-dimensional case) it considers all pairs and triples of locator circles.

References

1. Scarborough, James B., Numerical Mathematical Analysis, John Hopkins Press, Baltimore, 1930.
2. First Interim Report on NBS Project WHERE.

LISTING OF POSITION LOCATION PROGRAMS

BA RUN S00KNE,01563,1,30

189

09

QUIT FOR FINDIT,FINDIT
UNIVAC 1108 FORTRAN V LEVEL 2206 0018 F5018P
THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:42

SUBROUTINE FINDIT ENTRY POINT 001J20

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 001135
0000 *DATA 001121
0002 *BLANK 000000
0003 CBI 013000
0004 MXYZ 000673

EXTERNAL REFERENCES (BLOCK, NAME)

0005 LOCLSL
0006 LOCLSS
0007 HEIGHT
0010 LSS3D
0011 SQRT
0012 NERR3s

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0001	000150	1L	0001	000376	12L	0001	000027	127G	0001	000374	13L	0001	000601	14L
0001	000604	15L	0001	000627	16L	0001	000367	17L	0001	000660	19L	0001	000152	2L
0001	000715	20L	0001	000257	204G	0001	000212	22L	0001	000337	222G	0001	000403	241G
0001	000412	245G	0001	001074	25L	0001	000473	273G	0001	000253	3L	0001	000554	322G
0001	000524	34L	0001	000465	35L	0001	001015	36L	0001	000313	4L	0001	000751	401G
0001	001056	435G	0001	000442	5L	0001	000332	6L	0000	001056	900F	0003	R	011531 CLK
0000	R	001053 D	0003	R	000000 EP	0000	R	001046 EPX	0000	R	001047 EPY	0000	R	001044 F
0000	R	001036 FVAR	0003	L	011535 FXD	0000	R	001034 HMAX	0000	I	001042 I	0000	I	001032 IS
0000	I	001055 J	0003	I	011530 JFIND	0000	I	001043 K	0004	I	000001 L	0003	I	006116 LAM
0000	I	000341 LAMM	0000	I	001037 LFXD	0004	I	000000 M	0000	I	001054 MFXD	0000	I	001045 MM
0000	I	001050 N	0000	I	000471 NFXD	0000	I	001041 NPLANE	0003	I	012437 NTYPE	0000	I	001052 NWORST
0004	R	000672 P	0004	R	000542 R	0000	R	001033 RMAX	0003	L	012076 RPT	0003	R	006440 RR
0000	R	001035 RVAR	0003	L	007022 S	0000	L	000000 SI	0003	R	007363 TLOC	0003	R	011147 W
0000	R	001040 WOLD	0000	R	001051 WORST	0004	R	000412 WW	0003	R	011532 XNEW	0004	R	000002 XX
0003	R	011533 YNEW	0004	R	000132 YY	0003	R	011534 ZNEW	0004	R	000262 ZZ			

00100 1* C THIS ROUTINE ATTEMPTS TO LOCATE THE LOCATEE BY CALLING LSL OR THE
00100 2* C TWO- OR 3-DIMENSIONAL VERSIONS OR LSS.

00101 3* SUBROUTINE FINDIT

00103 4* PARAMETER LI=225, L4=L1+1

00104 5* LOGICAL S,SI(LI),RPT,FXD

00105 6* COMMON/CBI/EP(LI,14),LAM(L4),RR(L4),S(LI),TLOC(LI,4),W,L1),

00105 7* COMMON/CLK,XNEW,YNEW,ZNEW,FXD(LI),RPT(LI),NTYPE(LI)

00106 8* COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P

00107 9* DIMENSION LAMM(88),NFXD(LI)




```

00110 DATA S1,RMAX,HMAX,RVAR,FVAR,LFXD,NFXD/
00111 1 LI*.TRUE.,5.E6,200.,36.,40.,44.10/
00112 GOLD=1.E9
00121 IF(FXD(JFIND)).AND.W(JFIND).GT.O) WOLD=1./W(JFIND)-RVAR
00123 S1(JFIND)=S(JFIND)
00124 M=O
00125 NPLANE=O
00126 C COUNT THE LOCATORS. KEEPING TRACK OF THEIR POSITIONS. EIGHTS. AND
00127 C REPORTED DISTANCES TO THE LOCATEE. EXTRAPOLATE ON TWO PREVIOUS
00128 C POSITIONS TO FIND PRESENT POSITION OF EACH LOCATOR.
00129 DO 1 I=1,LI
00131 K=LAM(I)
00132 LAM(I)=K
00133 IF(RR(I).LT.O) GO TO 2
00135 IF(RR(I).GT.RMAX.OR..NOT.(S(K).AND.S1(K))) GO TO 1
00137 M=M+1
00140 LAM(M)=K
00141 RR(M)=RR(I)
00142 R(M)=RR(I)
00143 W(M)=W(K)
00144 F=O.O
00145 IF(TLOC(K,3).GT.TLOC(K,1).AND..NOT.FXD(K)) F=(CLK-TLOC(K,3))/
00146 1 (TLOC(K,3)-TLOC(K,1))
00147 XX(M)=EP(K,7)+F*(EP(K,7)-EP(K,1))
00150 YY(M)=EP(K,8)+F*(EP(K,8)-EP(K,2))
00151 ZZ(M)=EP(K,9)
00152 IF(NTYPE(K).LT.3) GO TO 1
00154 ZZ(M)=EP(K,9)+F*(EP(K,9)-EP(K,3))
00155 NPLANE=NPLANE+1
00156 1 CONTINUE
00160 2 F=O.
00161 ZNEA=EP(JFIND,9)
00162 IF(RPT(JFIND).AND.FXD(JFIND)) GO TO 22
00164 IF(M.LT.2) GO TO 5
00166 MM=1-1
00167 IF(.NOT.S(JFIND)) GO TO 3
00168 LAST ATTEMPT TO FIND THE LOCATEE WAS SUCCESSFUL. SO AN ESTIMATE
00169 OF THE PRESENT POSITION MAY BE FOUND BY EXTRAPOLATION.
00171 IF(TLOC(JFIND,3).GT.TLOC(JFIND,1)) F=(CLK-TLOC(JFIND,3))/
00172 1 (TLOC(JFIND,3)-TLOC(JFIND,1))
00173 22 XNEW=EP(JFIND,7)+F*(EP(JFIND,7)-EP(JFIND,1))
00174 YNEW=EP(JFIND,8)+F*(EP(JFIND,8)-EP(JFIND,2))
00175 A(JFIND)=FVAR
00176 IF(RPT(JFIND).AND.FXD(JFIND)) GO TO 19
00177 IF(NPLANE.GT.O.OR.NTYPE(JFIND).GT.2) GO TO 12
00178 C THE LOCATEE AND ALL LOCATORS ARE GROUND UNITS, SO LSS-20 IS USED.
00179 GO TO 6
00180 C THE LAST ATTEMPT TO LOCATE THE LOCATEE WAS NOT SUCCESSFUL. SO AN
00181 C ESTIMATE OF THE PRESENT POSITION MUST BE FOUND BY CALLING LSL.
00182 C FIRST, A SLANT-HEIGHT REDUCTION MUST BE DONE (3-D DISTANCES ARE
00183 C CONVERTED TO PLANAR DISTANCES).
00184 3 DO 4 I=1,M
00185 RR(I)=R(I)-.5*(ZNEW-ZZ(I))*2/R(I)
00186 IF(3.*(ZNEA-ZZ(I)).LT.R(I)) GO TO 4
00187 RR(I)=SQRT(ABS(R(I))*2-(ZNEW-ZZ(I))*2)
00188 4 CONTINUE
00189 CALL LCLSL
00190 IF(.NOT.S(JFIND)) GO TO 5
00191 67*

```



```

00217 68* IF(INTYPE(JFIND),GT,2) GO TO 13
00221 69* DO 7 J=1,M
00224 70* 7 RR(I)=R(I)-5*(ZNEW-ZZ(I))*2/R(I)
00226 71* S(JFIND)=FALSE.
00226 72* C LSS IS CALLED TO HOME IN ON THE POSITION OF A GROUND UNIT.
00227 73* CALL LOCLSS
00227 74* C WRITE(6,900) JFIND,S(JFIND),XNEW,YNEW,ZNEW,M
00230 75* 900 FORMAT(15,L3,3E14.8,15)
00231 76* IF(.NOT.S(JFIND)) GO TO 5
00231 77* C THE ALTITUDE OF THE GROUND UNIT IS FOUND.
00233 78* CALL HEIGHT(EPX,EPY,XNEW,YNEW,ZNEW,N)
00234 79* 17 IF (N.EQ.-1) GO TO 17
00236 80* GO TO 15
00236 81* C THE LOCATEE OR A LOCATOR IS AN AIRPLANE, AND LSS2D IS USED TO FIND
00236 82* C THE X- AND Y-COORDINATES MORE EXACTLY BEFORE CALLING LSS3D.
00237 83* 13 CALL LOCLSS
00237 84* C THE PROGRAM SEARCHES FARTHER AND FARTHER FROM THE LOCATEE UNTIL
00237 85* C AT LEAST 5 LOCATORS ARE FOUND.
00240 86* 12 DO 33 K=4,10,2
00243 87* L=0
00244 88* DO 32 I=1,M
00247 89* IF(K*(ZNEW-ZZ(I)).GT.R(I)) L=L+1
00251 90* 32 CONTINUE
00253 91* IF(L.GE.5) GO TO 35
00255 92* 33 CONTINUE
00257 93* IF(L.GE.4) GO TO 35
00257 94* C THE LOCATEE CANNOT BE FOUND.
00261 95* 5 S(JFIND)=FALSE.
00262 96* IF(.NOT.FXD(JFIND)) W(JFIND)=0.
00264 97* IF (.NOT.RPI(JFIND)) RETURN
00266 98* WORST=0.
00267 99* NWORST=JFIND
00270 100* RETURN
00270 101* C FIVE LOCATORS HAVE BEEN FOUND, SO LSS3D IS CALLED.
00271 102* 35 L=0
00272 103* DO 34 I=1,M
00275 104* IF(K*(ZNEW-ZZ(I)).LT.R(I)) GO TO 34
00277 105* L=L+1
00300 106* RR(I)=R(I)
00301 107* WW(I)=W(I)
00302 108* XX(I)=XX(I)
00303 109* YY(I)=YY(I)
00304 110* ZZ(I)=ZZ(I)
00305 111* LAM(I)=LAM(I)
00306 112* 34 CONTINUE
00310 113* M=L
00311 114* S(JFIND)=FALSE.
00312 115* CALL LSS3D
00313 116* IF(S(JFIND)) GO TO 15
00315 117* IF(INTYPE(JFIND),GT,2) GO TO 5
00315 118* C THE LOCATEE IS A GROUND UNIT, AND SOME OF THE LOCATORS (AT LEAST
00315 119* C ONE) ARE AIRPLANES. LSS3D COULD NOT FIND THE LOCATEE, SO SOME OF
00315 120* C THE LOCATORS ARE ELIMINATED, AND LSS2D IS USED.
00317 121* L=M
00320 122* M=0
00321 123* DO 14 I=1,L
00324 124* IF(4*(ZNEW-ZZ(I)).GT.R(I)) GO TO 14
00326 125* M=M+1

```



```

00327 126* XX(M)=XX(I)
00330 127* YY(M)=YY(I)
00331 128* ZZ(M)=ZZ(I)
00332 129* WW(M)=WW(I)
00333 130* R(M)=RR(I)
00334 131* 14 CONTINUE
00336 132* GO TO 6
00336 133* C THE LOCATEE HAS BEEN FOUND.
00337 134* 15 IF(FXD(JFIND)) GO TO 16
00337 135* C THE LOCATEE IS NOT A FIXED UNIT.
00341 136* EP(JFIND,1)=EP(JFIND,10)
00342 137* EP(JFIND,2)=EP(JFIND,11)
00343 138* EP(JFIND,3)=EP(JFIND,12)
00344 139* NFXD(JFIND)=0
00345 140* W(JFIND)=1./(W(JFIND)+RVAR)
00346 141* GO TO 20
00346 142* C THE LOCATEE IS A FIXED UNIT, SO A WEIGHTED AVERAGE OF ITS OLD AND
00346 143* C NEW POSITIONS IS CALCULATED, A WEIGHTED AVERAGE OF ITS OLD AND
00346 144* C NEW WEIGHTS IS ALSO CALCULATED.
00347 145* 16 D=WOLD+W(JFIND)
00350 146* XNEW=(EP(JFIND,4)*W(JFIND)+XNEW*WOLD)/D
00351 147* YNEW=(EP(JFIND,5)*W(JFIND)+YNEW*WOLD)/D
00352 148* ZNEW=(EP(JFIND,6)*W(JFIND)+ZNEW*WOLD)/D
00353 149* 19 WOLD=WOLD+NFXD(JFIND)*2
00354 150* NFXD(JFIND)=NFXD(JFIND)+1
00355 151* MFXD=NFXD(JFIND)*.2
00356 152* W(JFIND)=MFXD/(RVAR*MFXD+WOLD+W(JFIND))
00357 153* EP(JFIND,1)=XNEW
00360 154* EP(JFIND,2)=YNEW
00361 155* EP(JFIND,3)=ZNEW
00361 156* C 20 UPDATE THE ESTIMATED POSITION AND TIME OF LOCATION ARRAYS.
00362 157* EP(JFIND,4)=XNEW
00363 158* EP(JFIND,5)=YNEW
00364 159* EP(JFIND,6)=ZNEW
00365 160* EP(JFIND,7)=XNEW
00366 161* EP(JFIND,8)=YNEW
00367 162* EP(JFIND,9)=ZNEW
00370 163* EP(JFIND,10)=XNEW
00371 164* EP(JFIND,11)=YNEW
00372 165* EP(JFIND,12)=ZNEW
00373 166* EP(JFIND,13)=0.
00374 167* TLOC(JFIND,1)=TLOC(JFIND,4)
00375 168* TLOC(JFIND,2)=CLK
00376 169* TLOC(JFIND,3)=CLK
00377 170* TLOC(JFIND,4)=CLK
00377 171* C RELOCATE THE LOCATORS (THIS OPTION IS NOT NOW USED--SEE LSS3D).
00400 172* DO 36 I=1,L
00403 173* J=LAM(I)
00404 174* EP(J,7)=XX(I)
00405 175* EP(J,8)=YY(I)
00406 176* EP(J,9)=ZZ(I)
00407 177* EP(J,13)=EP(J,13)+1.0
00410 178* IF (EP(J,13).LT.EP(J,14)) GO TO 36
00412 179* EP(J,1)=EP(J,4)
00413 180* EP(J,2)=EP(J,5)
00414 181* EP(J,3)=EP(J,6)
00415 182* EP(J,4)=EP(J,7)
00416 183* EP(J,5)=EP(J,8)

```



```

00417 184* EP(J,6)=EP(J,9)
00420 185* EP(J,13)=0.0
00421 186* TLOC(J,1)=TLOC(J,2)
00422 187* TLOC(J,2)=TLOC(J,3)
00423 188* 36 TLOC(J,3)=CLK
00425 189* RETURN
00425 190* C IF YOU WANT THE REPORTING UNITS TO BE PRECISELY THOSE THAT HAVE
00425 191* C BEEN FIXED FOR TEN LOCATION OPERATIONS, REMOVE THE ABOVE RETURN
00425 192* C STATEMENT, AND REMOVE THE C FROM COLUMN 1 ON THE CARD BELOW.
00425 193* C RPT(JFIND)=NFXD(JFIND).GE.LFXD
00426 194* IF(RPT(JFIND).OR.W(JFIND).LE.WORST) RETURN
00430 195* RPT(NWORST)=.FALSE.
00431 196* RPT(JFIND)=.TRUE.
00432 197* WORST=W(JFIND)
00433 198* NWORST=JFIND
00434 199* DO 25 I=1,MM
00437 200* L=LAMM(I)
00440 201* IF(W(L).GT.WORST) GO TO 25
00442 202* WORST=W(L)
00443 203* NWORST=L
00444 204* 25 CONTINUE
00446 205* RETURN
00447 206* END

```

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

```

PHASE 1 TIME = 1 SEC.
PHASE 2 TIME = 0 SEC.
PHASE 3 TIME = 1 SEC.
PHASE 4 TIME = 0 SEC.
PHASE 5 TIME = 1 SEC.
PHASE 6 TIME = 0 SEC.

```

TOTAL COMPILATION TIME = 3 SEC

UNIVAC 1108 FORTRAN V LEVEL THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:45

SUBROUTINE LOCLSL ENTRY POINT 000464

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000501
0000 *DATA 003176
0002 *BLANK 000000
0003 CB1 013000
0004 MXYZ 000673

EXTERNAL REFERENCES (BLOCK, NAME)

0005 NWDUS
0006 N102\$
0007 NEXP1\$
0010 NERR3\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0001	000035	135G	0001	000044	141G	0001	000247	200G	0001	000333	216G	0001	000376	241G
0001	000173	3L	0001	000177	4L	0001	000352	8L	0000	003126	99F	0000	R	000000 A
0000	R	000620 B	0000	R	001440 C	0003	R	011531 CLK	0000	R	003122' CUTOFF	0000	R	003116 D
0000	R	003117 E	0000	R	003124 ELIM	0003	R	000000 EP	0000	R	003120 F	0003	R	011535 FXD
0003	I	011530 JFIND	0000	I	003115 K	0004	I	000001 L	0003	I	006116 LAM	0004	I	000000 M
0000	I	003102 MELIM	0000	I	003123 MLINE	0000	I	003125 N	0000	I	003114 NBEGIN	0000	I	003101 NELIM
0000	I	003104 NLINE	0000	I	003103 NN	0000	I	003113 NOFF	0003	I	012437 NTYPE	0004	L	000672 P
0004	R	000542 R	0003	R	012076 RPT	0003	R	006460 RR	0003	L	007022 S	0003	R	007343 TLOC
0000	R	003100 TOL	0000	R	003105 U1	0000	R	003106 U2	0000	R	003121 VAR	0000	R	002260 VRSQ
0000	R	003107 V1	0000	R	003110 V2	0003	R	011167 W	0004	R	000412 WW	0000	R	003111 W1
0000	R	003112 W2	0003	R	011532 XNEW	0004	R	000002 XX	0003	R	011533 YNEW	0004	R	000132 YY
0003	R	011534 ZNEW	0004	R	000262 ZZ									

00101 1* SUBROUTINE LOCLSL

00101 2* C
00101 3* C LEAST SQUARES LINEAR (LSL).
00101 4* C THIS ROUTINE MINIMIZES THE SUM OF THE SQUARED DISTANCES FROM THE
00101 5* C LOCATEE TO THE RADICAL AXIS LINES FORMED BY PAIRS OF CIRCLES.
00101 6* C LOCATEE TO THE RADICAL AXIS LINES FORMED BY PAIRS OF CIRCLES.
00101 7* C EACH CIRCLE HAS A LOCATOR FOR ITS CENTER, AND THE REPORTED
00101 8* C DISTANCE TO THE LOCATEE FOR ITS RADIUS. SEE FIRST INTERIM REPORT
00101 9* C ON PROJECT WHERSM, PP. 10-13.
00101 10* C THIS IS THE TWO-DIMENSIONAL ROUTINE.

00103 11* PARAMETER LI=225,L4=L1+1
00104 12* DIMENSION
00105 13* COMMON/CB1/EP(L1,14),LAM(L4),RR(L4),S(L1),TLOC(L1,4),W(L1),
00105 14* JFIND,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)


```

00106 15* COMMON/XYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P
00107 16* LOGICAL S,P
00110 17* DATA TOL/50./
00112 18* P=.TRUE.
00113 19* IF (P) WRITE(6,99) JFIND,CLK
00120 20* 99 FORMAT(' * * LSL2D: UNIT =15, * * CLOCK =F10.4)
00121 21* S(JFIND)=.FALSE.
00122 22* NELIM=4
00123 23* MELIM=0
00124 24* NN=M-1
00125 25* 2 NLINE=0
00126 26* U1=0.
00127 27* U2=0.
00130 28* V1=0.
00131 29* V2=0.
00132 30* W1=0.
00133 31* W2=0.
00134 32* DO 3 NOFF=1,NN
00137 33* NBEGIN=NOFF+1
00137 34* C FOR EACH PAIR OF CIRCLES, CALCULATE THE COEFFICIENTS OF THE
00137 35* C RADICAL AXIS LINE, A*X+B*Y-C=0, WHERE A=A+B*B=1. CALCULATE
00137 36* C COEFFICIENTS OF THE TWO EQUATIONS IN X AND Y, ON BOTTOM OF P. 13
00137 37* C OF FIRST INTERIM REPORT.
00140 38* DO 3 L=NBEGIN,M
00143 39* IF(NLINE.EQ.400) GO TO 4
00145 40* K=L-NOFF
00146 41* IF ((XX(K)-XX(L))*2+(YY(K)-YY(L))*2.LT.1.) GO TO 3
00150 42* NLINE=NLINE+1
00151 43* A(NLINE)=2.*(XX(K)-XX(L))
00152 44* B(NLINE)=2.*(YY(K)-YY(L))
00153 45* C(NLINE)=RR(L)*2-RR(K)*2+XX(K)*2-XX(L)*2+YY(K)*2-YY(L)*2
00154 46* D=A(NLINE)*A(NLINE)+B(NLINE)*B(NLINE)
00155 47* E=A(NLINE)/D
00156 48* F=B(NLINE)/D
00157 49* U1=U1+A(NLINE)*E
00160 50* V1=V1+B(NLINE)*E
00161 51* W1=W1+C(NLINE)*E
00162 52* U2=U2+A(NLINE)*F
00163 53* V2=V2+B(NLINE)*F
00164 54* W2=W2+C(NLINE)*F
00165 55* 3 CONTINUE
00170 56* 4 D=U1*V2-V1*U2
00171 57* IF(1.E6*ABS(D).LE.ABS(U1*V2)) RETURN
00171 58* C THE SYSTEM OF TWO EQUATIONS IS NON-SINGULAR, SOLVE FOR XNEW AND
00171 59* C CALCULATE SQUARED DISTANCES TO EACH RADICAL AXIS, AND AVERAGE.
00173 60* S(JFIND)=.TRUE.
00174 61* XNEW=(W1*V2-V1*W2)/D
00175 62* YNEW=(U1*W2-W1*U2)/D
00176 63* VAR=0.
00177 64* DO 6 M=1,NLINE
00202 65* VRSQ(M)=(A(M)*XNEW+B(M)*YNEW-C(M))*2/(A(M)*A(M)+B(M)*B(M))
00203 66* 6 VAR=VAR+VRSQ(M)
00205 67* VAR=VAR/FLOAT(NLINE)
00206 68* MELIM=MELIM+1
00207 69* IF(MELIM.GT.NELIM) RETURN
00211 70* CUTOFF=2.*(4-MELIM)
00212 71* NLINE=NLINE
00213 72* NLINE=0

```


00214	73*	ELIM=CUTOFF*VAR
00214	74*	C ELIMINATE RADICAL AXES WHOSE SQUARED DISTANCE, VRSQ, IS GREATER
00214	75*	C THAN CUTOFF TIMES THE AVERAGE, WHERE CUTOFF IS, SUCCESSIVELY,
00214	76*	C 8, 4, 2, AND 1.
00215	77*	DO 8 N=1,MLINE
00220	78*	IF(VRSQ(N).GT.ELIM) GO TO 8
00222	79*	MLINE=MLINE+1
00223	80*	A(NLINE)=A(N)
00224	81*	B(NLINE)=B(N)
00225	82*	C(NLINE)=C(N)
00226	83*	B CONTINUE
00230	84*	IF(NLINE.LE.2) RETURN
00232	85*	U1=0.
00233	86*	U2=0.
00234	87*	V1=0.
00235	88*	V2=0.
00236	89*	W1=0.
00237	90*	W2=0.
00237	91*	C RECALCULATE COEFFICIENTS OF THE TWO LINEAR EQUATIONS.
00240	92*	DO 9 N=1,NLINE
00243	93*	D=A(N)*A(N)+B(N)*B(N)
00244	94*	E=A(N)/D
00245	95*	F=B(N)/D
00246	96*	U1=U1+A(N)*E
00247	97*	V1=V1+B(N)*E
00250	98*	W1=W1+C(N)*E
00251	99*	U2=U2+A(N)*F
00252	100*	V2=V2+B(N)*F
00253	101*	9 W2=W2+C(N)*F
00255	102*	GO TO 4
00256	103*	END

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

PHASE 1 TIME =	1 SEC.
PHASE 2 TIME =	0 SEC.
PHASE 3 TIME =	0 SEC.
PHASE 4 TIME =	0 SEC.
PHASE 5 TIME =	1 SEC.
PHASE 6 TIME =	0 SEC.

TOTAL COMPILATION TIME = 2 SEC

WIT FOR LOCLSS, LOCLSS

UNIVAC 1108 FORTRAN V LEVEL 2206 0018 F501AP
THIS COMPILATION WAS DONE ON 05 MAY 72 AT 11:41:47

SUBROUTINE LOCLSS ENTRY POINT 000276

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000636
0000 *DATA 000300
0002 *BLANK 000000
0003 C81 013000
0004 C83 002261
0005 MXYZ 000673

EXTERNAL REFERENCES (BLOCK, NAME)

0006 NEXPIs
0007 NERR3s

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0001	000010	IL	0001	000016	132G	0001	000073	151G	0001	000512	21	0001	000226	230G
0001	000220	25L	0001	000324	254G	0001	000551	322G	0001	000032	4L	0001	000155	5L
0001	000203	8L	0000	000170	80F	0000	000173	81F	0000	000175	82F	0003	R 011531	CLK
0000	R 000166	D	0000	R 000157	DELXSQ	0000	R 000160	DELYSQ	0000	R 000153	DFDX	0000	R 000155	DFDY
0000	R 000162	DGDX	0000	R 000154	DGDY	0000	R 000161	DSQ	0000	R 000165	DSQ	0000	R 000163	DX
0000	R 000164	DY	0000	R 000146	E	0000	R 000143	EAVE	0000	R 000000	EE	0000	R 000142	ELIM
0003	R 000000	EP	0000	R 000151	F	0000	R 000167	F	0000	R 000132	FRAC	0003	R 011535	FXD
0000	R 000152	G	0000	I 000144	I	0000	I 000141	I1	0004	I 000000	IN	0004	I 002260	I9
0000	I 000145	J	0003	I 011530	JFIND	0000	I 000135	K	0005	I 000001	L	0003	I 006116	LAM
0000	I 000134	LR	0005	I 000000	M	0000	I 000140	MELIM	0000	I 000136	MRELOC	0000	I 000130	NELIM
0000	I 000131	NRELOC	0003	I 012437	NTYPE	0000	R 000137	OLDEAV	0005	L 000672	P	0005	R 000542	R
0003	R 012076	RPT	0003	R 006460	RR	0000	R 000150	RSQ	0003	L 007022	S	0000	L 000133	STOP
0003	R 007363	TLOC	0000	R 000156	TW	0000	R 000147	V	0003	R 011167	W	0005	R 000412	WW
0003	R 011532	XNEW	0005	R 000002	XX	0003	R 011533	YNEW	0005	R 000132	YY	0003	R 011534	ZNEW
0005	R 000262	ZZ												

00101 1* SUBROUTINE LOCLSS

00101 2* C THIS IS THE TWO-DIMENSIONAL LEAST SQUARES SQUARED ROUTINE. IT
00101 3* C ATTEMPTS TO MINIMIZE THE FUNCTION EAVE BY FINDING A POINT WHERE
00101 4* C ITS DERIVATIVES F AND G WITH RESPECT TO X AND Y ARE ZERO. TO DO
00101 5* C THIS, IT USES NEWTON-RAPHSON ITERATION.

00101 6* C

00103 7* PARAMETER L1=225, L4=L1+1

00104 8* COMMON /CB1/EP(L1,14), LAM(L41,RR(L4),S(L1),TLOC(L1+4),W(L1),

00104 9* JFIND,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)

00105 10* COMMON /CB3/IN(600,2),I9

00106 11* COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P

00107 12* DIMENSION EE(88)


```

00000 100 DATA NELIM,MRELOC,MAC/1,0,0,1/
00114 14 LOGICAL S,STOP
00115 15 LOGICAL P
00116 16 DATA P/,FALSE,/
00120 17 IN(19,1)=0
00121 18 IN(19,2)=0
00122 19 LR=0
00123 20 L=M
00124 21 K=0
00125 22 MRELOC=0
00126 23 I STOP=,FALSE,
00127 24 OLDEAV=1,E30
00130 25 MELIM=0
00130 26 C DO UP TO FOUR NEWTON-RAPHSON ITERATIONS. STOPPING IF CONVERGENCE
00130 27 C OR DIVERGENCE OCCURS.
00131 28 DO 2 I=0,3
00134 29 IF (STOP) GO TO 4
00136 30 CALL NEWTON
00137 31 IF (S(JFIND)) GO TO 4
00141 32 2 CONTINUE
00143 33 4 MELIM=MELIM+1
00144 34 K=0
00145 35 IF (MELIM.GT.MELIM) GO TO 8
00145 36 C ELIMINATE ANY LOCATOR I SUCH THAT EE(I) IS TOO BIG. PUT
00145 37 C ELIMINATED LOCATORS AT THE BOTTOM OF THE LIST.
00147 38 ELIM=EAVE+2*(3-MELIM)
00150 39 DO 5 I=L,1,-1
00153 40 IF (EE(I).LE.ELIM) GO TO 5
00155 41 K=K+1
00156 42 J=L+1-K
00157 43 IF (I.EQ.J) GO TO 5
00161 44 E=WR(J)
00162 45 WW(J)=WW(I)
00163 46 WA(I)=E
00164 47 E=XX(J)
00165 48 XX(J)=XX(I)
00166 49 XX(I)=E
00167 50 E=YY(J)
00170 51 YY(J)=YY(I)
00171 52 YY(I)=E
00172 53 E=ZZ(J)
00173 54 ZZ(J)=ZZ(I)
00174 55 ZZ(I)=E
00175 56 E=R(J)
00176 57 R(J)=R(I)
00177 58 R(I)=E
00200 59 E=RR(J)
00201 60 RR(J)=RR(I)
00202 61 RR(I)=E
00203 62 I=LAM(J)
00204 63 LAM(J)=LAM(I)
00205 64 LAM(I)=I
00206 65 5 CONTINUE
00210 66 IN(19,MRELOC+1)=IN(19,MRELOC+1)+K
00210 67 C DO NOT ELIMINATE MORE THAN ONE FOURTH OF THE LOCATORS.
00211 68 IF (4*(L-K).LT.3*M) GO TO 25
00213 69 L=L-K
00214 70 IF (K) 4,4,1

```



```

00217 71*      8 MRELOC=MRELOC+1
00220 72*      IF (MRELOC,GT,MRELOC) GO TO 25
00220 73*      C      RELOCATE THE LOCATORS.  IF NRELOC=0 (SEE DATA STATEMENTS), THIS
00220 74*      C      OPTION IS NOT USED.
00222 75*      LR=L
00223 76*      CALL RELOC
00224 77*      MELIM=0
00225 78*      GO TO 1
00225 79*      C      CALCULATE VARIANCE OF THE POSITION LOCATION ESTIMATE.
00226 80*      25 V=0.
00227 81*      DO 26 I=1,L
00228 82*      RSQ=RR(I)*.2
00229 83*      26 V=V+(RSQ-((XNEW-XX(I))*2+(YNEW-YY(I))*2))*2/RSQ
00230 84*      W(JFIND)=V/(4*L-8)
00231 85*      L=L-R
00232 86*      RETURN
00233 87*      C      NEWTON-RAPHSON ITERATION.  CALCULATE F, G, AND THEIR DERIVATIVES.
00234 88*      C      SOLVE FOR DX AND DY, AND INCREMENT XNEW AND YNEW.
00235 89*      C      SUBROUTINE NEWTON
00236 90*      F=0.
00237 91*      G=0.
00238 92*      DFDX=0.
00239 93*      DGDY=0.
00240 94*      DFDY=0.0
00241 95*      EAVE=0.
00242 96*      C      IF (P) WRITE(6,80)
00243 97*      80 FORMAT(' R,R2,D2,E,')
00244 98*      TW=0.
00245 99*      DO 1 I=1,L
00246 100*      RSQ=RR(I)*.2
00247 101*      DELXSQ=(XNEW-XX(I))*2/RSQ
00248 102*      DELYSQ=(YNEW-YY(I))*2/RSQ
00249 103*      DSQ=(DELXSQ+DELYSQ)*RSQ
00250 104*      E=DELXSQ-1.+DELYSQ
00251 105*      EE(I)=E+E*RSQ
00252 106*      EAVE=EAVE+EE(I)*WW(I)
00253 107*      CC      IF (P) WRITE(6,81) RR(I),RSQ,DSQ,EE(I)
00254 108*      81 FORMAT(9E14.8)
00255 109*      TW=TW+WW(I)
00256 110*      E=F+E*(XNEW-XX(I))*WW(I)
00257 111*      G=G+E*(YNEW-YY(I))*WW(I)
00258 112*      DFDX=DFDX+(E+2.*DELXSQ)*WW(I)
00259 113*      DGDY=DGDY+(E+2.*DELYSQ)*WW(I)
00260 114*      1 DFDY=DFDY+2.*(XNEW-XX(I))*WW(I)/RSQ
00261 115*      EAVE=EAVE/TW
00262 116*      IF (EAVE,GT,OLDEAV) GO TO 2
00263 117*      OLDEAV=EAVE
00264 118*      DGDY=DFDY
00265 119*      E=DFDX*DGDY-DFDY*DGDX
00266 120*      IF (1.E6*ABS(E).LE.ABS(DFDX*DGDY)) GO TO 2
00267 121*      DX=(DFDY*G-DGDY*F)/E
00268 122*      DY=(DGDY*F-DFDX*G)/E
00269 123*      XNEW=XNEW+DX
00270 124*      YNEW=YNEW+DY
00271 125*      SJFIND)=AMAX1(ABS(F),ABS(G),LT,TW
00272 126*      C      IF (PJ) WRITE(6,82) DX,DY,XNEW,YNEW,E,G,EAVE
00273 127*      82 FORMAT(' AX,AY,XNEW,YNEW,DE/DX,DE/DY,EAVE/8E14.8)
00274 128*      RETURN

```



```

00313 129* C THE ITERATION DIVERGED. DO NOT CHANGE XNEW AND YNEW.
00314 130* 2 STOP=,TRUE,
00315 131* RETURN
00315 132* C RELOCATE THE LOCATORS,
00316 133* SUBROUTINE RELOC
00321 134* DO 1 I=1,L
00324 135* DSQ=(XNEW-XX(I))*2+(YNEW-YY(I))*2
00325 136* D=RR(I)
00326 137* D=.5*(D+DSQ/D)
00327 138* D=.5*(D+DSQ/D)
00330 139* F=FRAC*(D-RR(I))/D
00331 140* XX(I)=XX(I)+F*(XNEW-XX(I))
00332 141* YY(I)=YY(I)+F*(YNEW-YY(I))
00333 142* 1 ZZ(I)=ZZ(I)+F*(ZNEW-ZZ(I))
00335 143* RETURN
00336 144* END

```

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(5)

```

PHASE 1 TIME = 0 SEC.
PHASE 2 TIME = 1 SEC.
PHASE 3 TIME = 0 SEC.
PHASE 4 TIME = 0 SEC.
PHASE 5 TIME = 1 SEC.
PHASE 6 TIME = 0 SEC.

```

TOTAL COMPILATION TIME = 2 SEC

SUBROUTINE LSS3D ENTRY POINT 000310

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 001216
0000 *DATA 000322
0002 *BLANK 000000
0003 CB1 013000
0004 CB3 002261
0005 MXYZ 000673

EXTERNAL REFERENCES (BLOCK, NAME)

0006 NEXPI\$
0007 NERR3\$
0010 NWDUS
0011 NI02\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0001	001046	JL	0001	000010	IL	000022	J33G	0001	000102	143G	0001	000723	2L		
0001	000234	231G	0001	000226	25L	000341	260G	0001	001120	375G	0001	000040	4L		
0001	000164	5L	0001	001062	5L	000212	8L	0000	000214	9F	0000	000204	99F		
0000	R	000200	ALPHA	0003	R	001531	CLK	0000	R	000202	D	0000	R	000170	DELXSQ
0000	R	000171	DELYSQ	0000	R	000172	DELZSQ	0000	R	000174	DETINV	0000	R	000160	DFDX
0000	R	000161	DFDY	0000	R	000162	DFDZ	0000	R	000164	DGDZ	0000	R	000165	DHDZ
0000	R	000201	DSQ	0000	R	000175	DX	0000	R	000177	DZ	0000	R	000152	E
0000	R	000147	EAVE	0000	R	000000	EE	0000	R	000000	EP	0000	R	000141	EX
0000	R	000142	EY	0000	R	000143	EZ	0000	R	000203	F	0000	R	000133	FRAC
0003	R	011535	FXD	0000	R	000156	G	0000	R	000155	F	0000	R	000144	I1
0004	I	000000	IN	0004	I	002260	I9	0000	I	000157	H	0000	I	000135	K
0005	I	000001	L	0003	I	006116	LAM	0000	I	000151	J	0000	I	000137	MELIM
0000	I	000136	MRELOC	0000	I	000131	NELIM	0000	I	000134	LR	0000	I	000140	OLDEAV
0005	L	000672	P	0005	R	000542	R	0003	R	012076	RPT	0003	R	000154	RSQ
0000	R	000167	RSQINV	0003	L	007022	S	0000	L	000130	STOP	0003	R	000166	TW
0000	R	000153	V	0003	R	011167	W	0005	R	000412	WW	0003	R	011532	XNEW
0003	R	011533	YNEW	0005	R	000132	YY	0003	R	011534	ZNEW	0005	R	000002	XX

00101 1* SUBROUTINE LSS3D
00101 2* C THIS SUBROUTINE IS THE 3-DIMENSIONAL LEAST SQUARES SQUARED METHOD.
00101 3* C IT MINIMIZES A FUNCTION, EAVE, BY TRYING TO GET ITS 3 DERIVATIVES
00101 4* C (F,G,H RESPECTIVELY) WITH RESPECT TO X, Y, AND Z TO BE ZERO.
00101 5* C NEWTON-RAPHSON ITERATION IS USED TO APPROACH THE ZEROES.
00101 6* C
00103 7* C
00104 8* C
PARAMETER LI=225, L4=L1+1
COMMON/CB1/EP(L1,14),LAM(L4),RR(L4),S(L1),TLOC(L1,4),W(L1)


```
00105 10* COMMON /CB3/IN(600,2),I9
00106 11* COMMON/MXYZ/M,L,XX(88),YY(88),ZZ(88),WW(88),R(88),P
00107 12* DIMENSION EE(88)
00110 13* LOGICAL S,STOP,P
00111 14* DATA NELIM,NKELOC,FRAC/I,0,,2/
00111 15* IF (P) WRITE(6,99) JFIND,CLK
00115 16* C 99 FORMAT(' * * L553D: UNIT =',I5,' CLOCK =',F10.4)
00116 17* IN(I9,I)=0
00117 18* IN(I9,2)=0
00120 19* LR=0
00121 20* L=M
00122 21* K=0
00123 22* MRELOC=0
00124 23* 1 STOP=.FALSE.
00125 24* MELIM=0
00126 25* OLDEAV=1.E30
00127 26* EX=10000.
00130 27* EY=10000.
00131 28* EZ=10000.
00131 29* C DO UP TO FOUR ITERATIONS, STOPPING IF CONVERGENCE OCCURS, OR IF
00131 30* C THE FIRST ITERATION DIVERGES.
00132 31* DO 2 II=0,3
00135 32* IF(.STOP) GO TO 4
00137 33* CALL NEWTON(II,K)
00140 34* IF(S(JFIND)) GO TO 4
00142 35* 2 CONTINUE
00144 36* 4 MELIM=MELIM+1
00145 37* K=0
00146 38* IF(MELIM.GT.NELIM) GO TO 8
00150 39* CUTOFF=2*(3-MELIM,
00151 40* ELIM=CUTOFF*EAVE
00151 41* C ELIMINATE ANY LOCATOR I SUCH THAT EE(I) IS TOO BIG. PUT
00151 42* C ELIMINATED LOCATORS AT THE BOTTOM OF THE LIST.
00152 43* DO 5 I=L,I,-1
00155 44* IF(EE(I).LE.ELIM) GO TO 5
00157 45* K=K+1
00160 46* J=L+1-K
00161 47* IF(I.EQ.J) GO TO 5
00163 48* E=WA(J)
00164 49* WW(J)=WW(I)
00165 50* WA(I)=E
00166 51* E=XX(J)
00167 52* XX(J)=XX(I)
00170 53* XX(I)=E
00171 54* E=YY(J)
00172 55* YY(J)=YY(I)
00173 56* YY(I)=E
00174 57* E=ZZ(J)
00175 58* ZZ(J)=ZZ(I)
00176 59* ZZ(I)=E
00177 60* E=R(J)
00200 61* R(J)=R(I)
00201 62* R(I)=E
00202 63* E=RR(J)
00203 64* RR(J)=RR(I)
00204 65* RR(I)=E
00205 66* II=LAM(J)
```



```

00206 67* LAM(J)=LAM(I)
00207 68* LAM(I)=11
00210 69* 5 CONTINUE
00212 70* IN(I9,MRELOC+1)=IN(I9,MRELOC+1)*K
00212 71* C DO NOT ELIMINATE MORE THAN ONE FOURTH OF THE LOCATORS.
00213 72* IF(4*(L-K).LI.3*M) GO TO 25
00215 73* L=L-K
00216 74* IF(K) 4,4,1
00221 75* 8 MRELOC=MRELOC+1
00222 76* IF(MRELOC.GT.NRELOC) GO TO 25
00222 77* C RELOCATE THE LOCATORS. IF NRELOC=0 (SEE DATA STATEMENTS), THIS
00222 78* C OPTION IS NOT NOW USED.
00224 79* LR=L
00225 80* CALL RELOC
00226 81* GO TO 1
00226 82* C CALCULATE VARIANCE OF THE POSITION ESTIMATE.
00227 83* 25 V=0.
00230 84* DO 26 I=1,L
00233 85* RSQ=RR(I)**2
00234 86* 26 V=V+(RSQ-((XNEW-XX(I))**2+(YNEW-YY(I))**2+(ZNEW-ZZ(I))**2))/RSQ
00236 87* W(JFIND)=V/(4*L-12)
00237 88* L=LR
00240 89* RETURN
00240 90* C NEWTON RAPHSON ITERATION. CALCULATE F, G, H AND THEIR DERIVATIVES
00240 91* C AND FIND DX, DY, DZ BY SOLVING EQUATIONS (5) OF WORKING PAPER NO.
00240 92* C 7 OF THE FINAL REPORT (MAY 1972).
00241 93* SUBROUTINE NEWTON(I,K)
00244 94* F=0.
00245 95* G=0.
00246 96* H=0.
00247 97* DFDX=0.
00250 98* DFDY=0.
00251 99* DFDZ=0.
00252 100* DGDY=0.
00253 101* DGDZ=0.
00254 102* DHDZ=0.
00255 103* EAVE=0.
00256 104* TW=0.
00257 105* DO 1 I=1,L
00262 106* RSQ=RR(I)**2
00263 107* RSQINV=1./RSQ
00264 108* DELXSQ=RSQINV*(XNEW-XX(I))**2
00265 109* DELYSQ=RSQINV*(YNEW-YY(I))**2
00266 110* DELZSQ=RSQINV*(ZNEW-ZZ(I))**2
00267 111* E=DELXSQ+DELYSQ-1.+DELZSQ
00270 112* EE(I)=E+E*RSQ
00271 113* EAVE=EAVE+EE(I)*WW(I)
00272 114* TW=TW+WW(I)
00273 115* F=F+E*(XNEW-XX(I))*WW(I)
00274 116* G=G+E*(YNEW-YY(I))*WW(I)
00275 117* H=H+E*(ZNEW-ZZ(I))*WW(I)
00276 118* DFDX=DFDX+(E+2.*DELXSQ)*WW(I)
00277 119* DFDY=DFDY+2.*(XNEW-XX(I))*(YNEW-YY(I))*RSQINV*WW(I)
00300 120* DFDZ=DFDZ+2.*(XNEW-XX(I))*(ZNEW-ZZ(I))*RSQINV*WW(I)
00301 121* DGDY=DGDY+(E+2.*DELYSQ)*WW(I)
00302 122* DGDZ=DGDZ+2.*(YNEW-YY(I))*(ZNEW-ZZ(I))*RSQINV*WW(I)
00303 123* 1 DHDZ=DHDZ+(E+2.*DELZSQ)*WW(I)
00305 124* EAVE=EAVE/TW

```



```

00306 125* IF (EAVE,GT,OLDEAV) GO TO 2
00310 126* DET=DFDX*(DGDY,DHDZ,DGDZ**2)+DFDY*(DGDZ*DEQZ-DFDY*DHDZ,
00310 127* 1 +DFDZ*(DFDY*DGDZ-DGDY*DFDZ)
00311 128* IF (ABS(DET),LT,1.E-8) GO TO 2
00313 129* DETINV=1./DET
00314 130* DX=DETINV*(F*(DGDZ**2-DGDY*DHDZ)+G*(DFDY*DHDZ-DFDZ*DGDZ))+
00314 131* 1 H*(DFDZ*DGDY-DFDY*DGDZ))
00315 132* DY=DETINV*(F*(DFDY*DHDZ-DGDZ*DEQZ)+G*(DEQZ**2-DFDX*DHDZ))+
00315 133* 1 H*(DFDX*DGDZ-DFDZ*DFDY))
00316 134* DZ=DETINV*(F*(DGDY*DFDZ-DFDY*DGDZ)+G*(DFDX*DGDZ-DFDY*DEQZ))+
00316 135* 1 H*(DFDY**2-DFDX*DGDY))
00317 136* OLDEAV=EAVE
00317 137* C IF (P)WRITE(6,9)F,G,H,DFDX,DFDY,DFDZ,DGDZ,DHDZ,XNEW,YNEW,ZNEW,
00317 138* C 1 DX,DY,DZ,(XX(I),I=1,M),(YY(I),I=1,M),(ZZ(I),I=1,M),(R(I),I=1,M),
00317 139* C 2(WW(I),I=1,M),(EE(I),I=1,M)
00320 140* 9 FORMAT(1X,15E6,3)
00320 141* C IF THIS STEP SIZE IS NOT MUCH SMALLER THAN THAT OF THE PREVIOUS
00320 142* C ITERATION, CORRECT THE SIZE OF THIS ITERATION.
00321 143* C CALL CORRECT(IX,DX)
00322 144* C CALL CORRECT(EY,DY)
00323 145* C CALL CORRECT(EZ,DZ)
00324 146* XNEW=XNEW+DX
00325 147* YNEW=YNEW+DY
00326 148* ZNEW=ZNEW+DZ
00327 149* S(JFIND)=AMAX1(ABS(F),ABS(G),ABS(H)),LT,TW
00330 150* RETURN
00330 151* C THE ITERATION HAS DIVERGED. IF THIS IS THE FIRST ITERATION FOR
00330 152* C THIS LOCATEE, THE ROUTINE CONKS OUT.
00331 153* 2 STOP=,TRUE.
00332 154* WRITE(6,9) EAVE,OLDEAV,DET
00337 155* IF (J1.EQ.O.AND.K.EQ.O) RETURN
00337 156* C DIVERGENCE FOR SOME OTHER ITERATION (NOT THE FIRST). HALVE THE
00337 157* C STEP SIZE OF THE PREVIOUS ITERATION, AND RETURN TO TRY ANOTHER
00337 158* C ITERATION.
00341 159* STOP=,FALSE.
00342 160* DX=DX/2.
00343 161* DY=DY/2.
00344 162* DZ=DZ/2.
00345 163* XNEW=XNEW-DX
00346 164* YNEW=YNEW-DY
00347 165* ZNEW=ZNEW-DZ
00350 166* RETURN
00351 167* SUBROUTINE CORRECT(A,B)
00354 168* IF (ABS(B/A).LT,.5) GO TO 5
00356 169* ALPHA=B/A
00357 170* IF (ALPHA.GT.O.)GO TO 1
00361 171* B=B/(1.-ALPHA)
00362 172* GO TO 5
00363 173* 1 A=2.
00364 174* IF (ALPHA.LT.1.) A=2.*ALPHA
00366 175* B=B*A
00367 176* 5 A=B
00370 177* RETURN
00370 178* C RELOCATE THE LOCATORS.
00371 179* SUBROUTINE RELOC
00374 180* DO 1 I=1,L
00377 181* DSQ=(XNEW-XX(I))*2+(YNEW-YY(I))*2+(ZNEW-ZZ(I))*2
00400 182* D=RR(I)

```


00401	183*	D=.5*(D+DSQ/D)
00402	184*	D=.5*(D+DSQ/D)
00403	185*	D=.5*(D+DSQ/D)
00404	186*	F=FRAC*(D-RR(I))/D
00405	187*	XX(I)=XX(I)+F*(XNEW-XX(I))
00406	188*	YY(I)=YY(I)+F*(YNEW-YY(I))
00407	189*	1 ZZ(I)=ZZ(I)+F*(ZNEW-ZZ(I))
00411	190*	RETURN
00412	191*	END

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

PHASE 1 TIME =	1 SEC.
PHASE 2 TIME =	0 SEC.
PHASE 3 TIME =	0 SEC.
PHASE 4 TIME =	0 SEC.
PHASE 5 TIME =	1 SEC.
PHASE 6 TIME =	0 SEC.

TOTAL COMPILATION TIME = 2 SEC

•• CONSOLE TYPEOUTS ••

TIME	TYPE	MESSAGE
11:41:42	E	## 205 RUN 11:41:42 *****
11:41:42	E	A-09
11:41:42	E	SOOKNE,01563,1,30
11:41:48	E	114147 GLADNE 130 MNT @0600Z
11:41:52	E	I: 649 C: OP: 21LP: 818 E:0 0 10 L: 189
11:41:52	E	418 COUNT IS 9
11:41:54	E	ZFILE COUNT IS 1

TAPE READS	0,RETRYs	0
TAPE WRITES	0,RETRYs	0

05 MAY 72 11:41:42 TO 11:41:52 IDENT:SOOKNE ACCOUNT:01563 CARDS IN: 649 CARDS OUT: 0 PAGES: 21 ELAPSE:n 0 10 LOG 189
RUN ENTERED SYSTEM AT 11:41:37 PRIORITY: A PRINT CHANNEL: 14 PUNCH CHANNEL: 13 LINES PRINTED: 818 ID: 09 SCRATCH: 0 USER 0

.. NBS EXEC II LEVEL 6.0A MOD 5.0 REEL 7368 *** MAY 2 , 1972 **

***** FORTAN COMPILER ***** 49 *****

FIXES HAVE BEEN INSERTED IN THE FORTAN COMPILER LEVEL 25. TESTING CONTINUES
ON THIS REVISED COMPILER. THE OPERATIONAL DATE OF LEVEL 25 WILL BE ANNOUNCED.

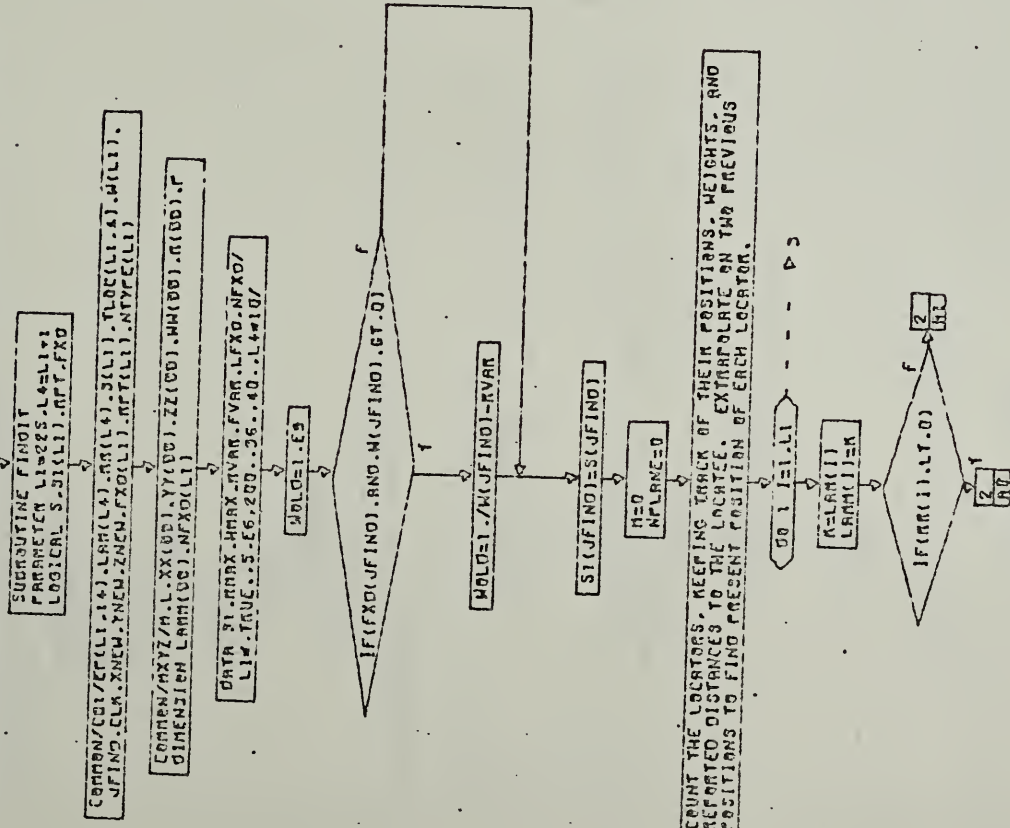
***** MAY 4, 1972 *****

.. .. NBS EXEC II LEVEL 6.0A MOD 5.0 REEL 7368 *** MAY 2 , 1972

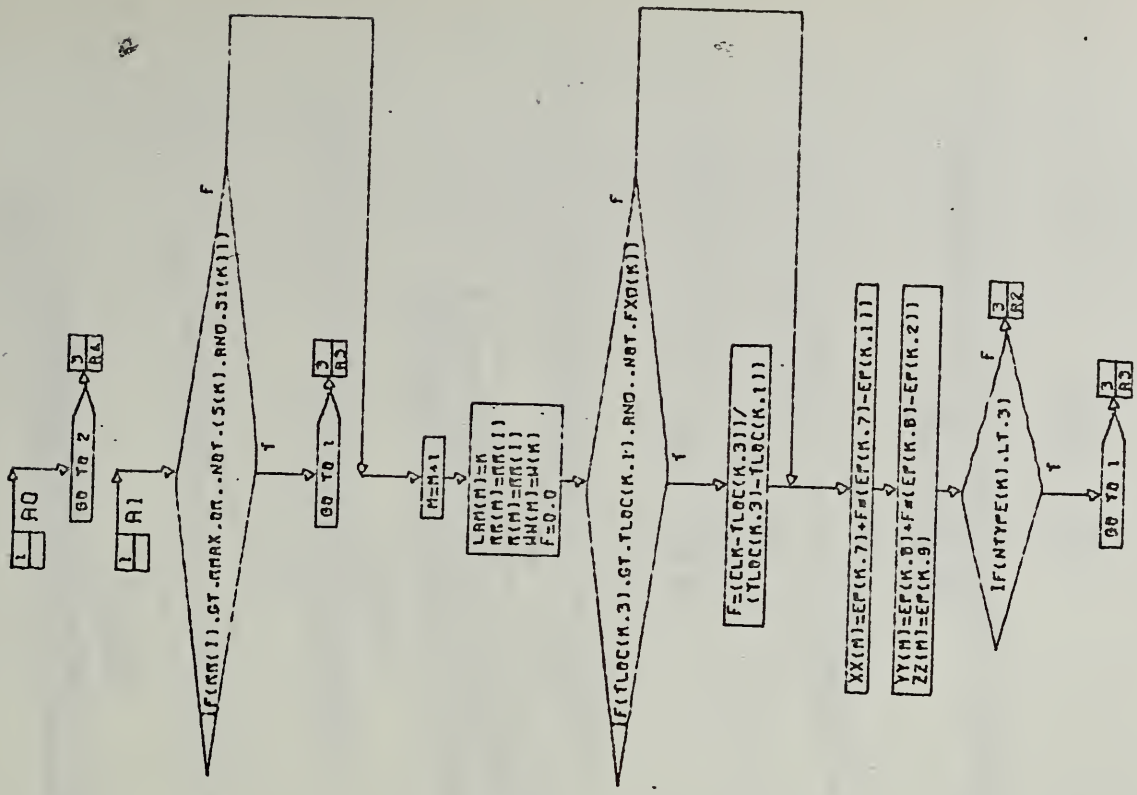
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

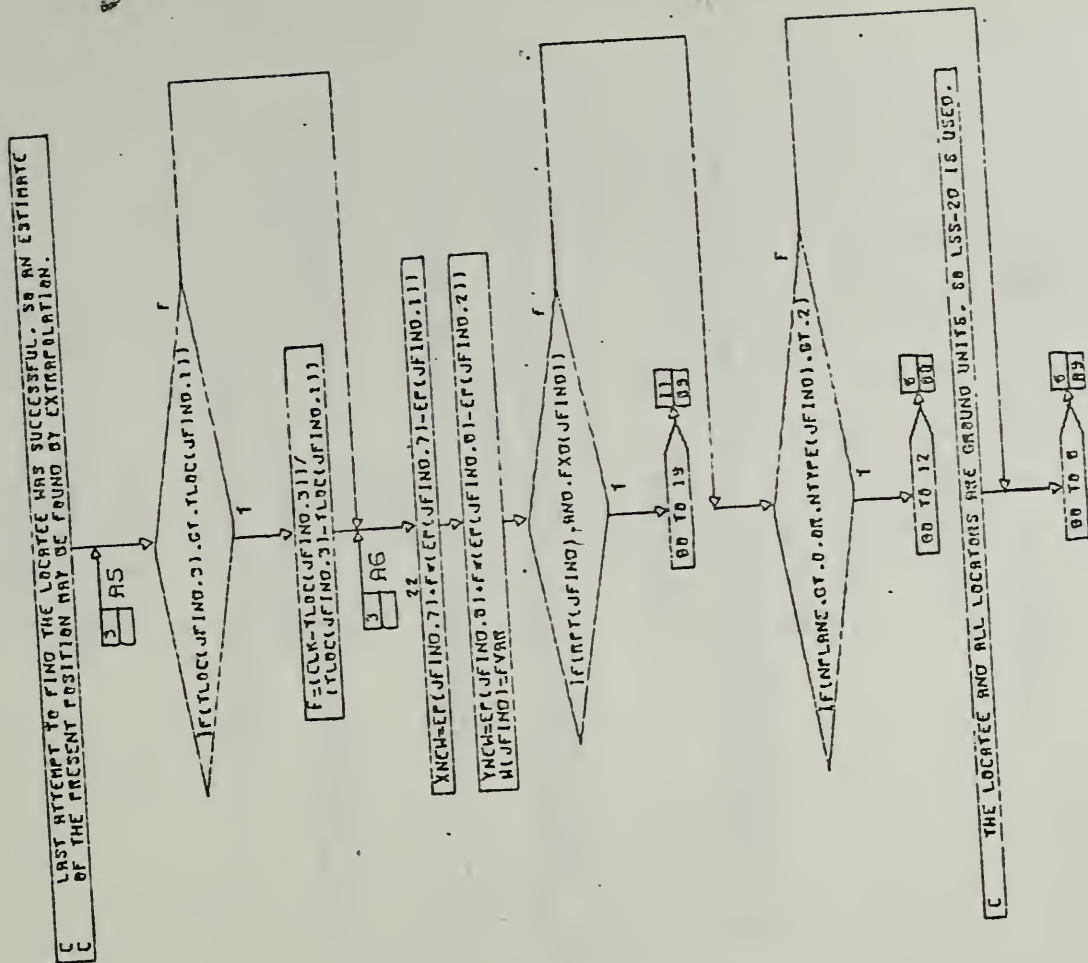
FLOW CHARTS FOR POSITION LOCATION PROGRAMS

C THIS ROUTINE ATTEMPTS TO LOCATE THE LOCATEE BY CALLING LSL OR THE TWO- OR 3-DIMENSIONAL VERSIONS OR LSE.



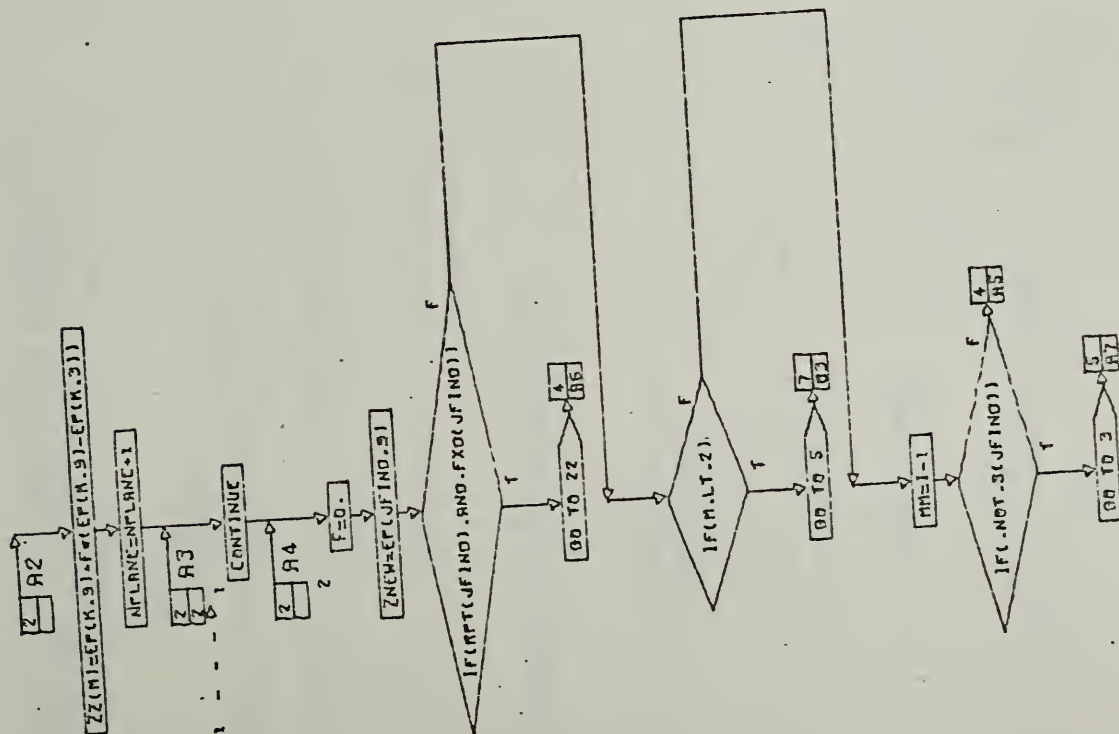
C COUNT THE LOCATORS, KEEPING TRACK OF THEIR POSITIONS, WEIGHTS, AND REPORTED DISTANCES TO THE LOCATEE. EXTRAPOLATE ON TWO PREVIOUS POSITIONS TO FIND PRESENT POSITION OF EACH LOCATOR.





CONT. ON P0 5

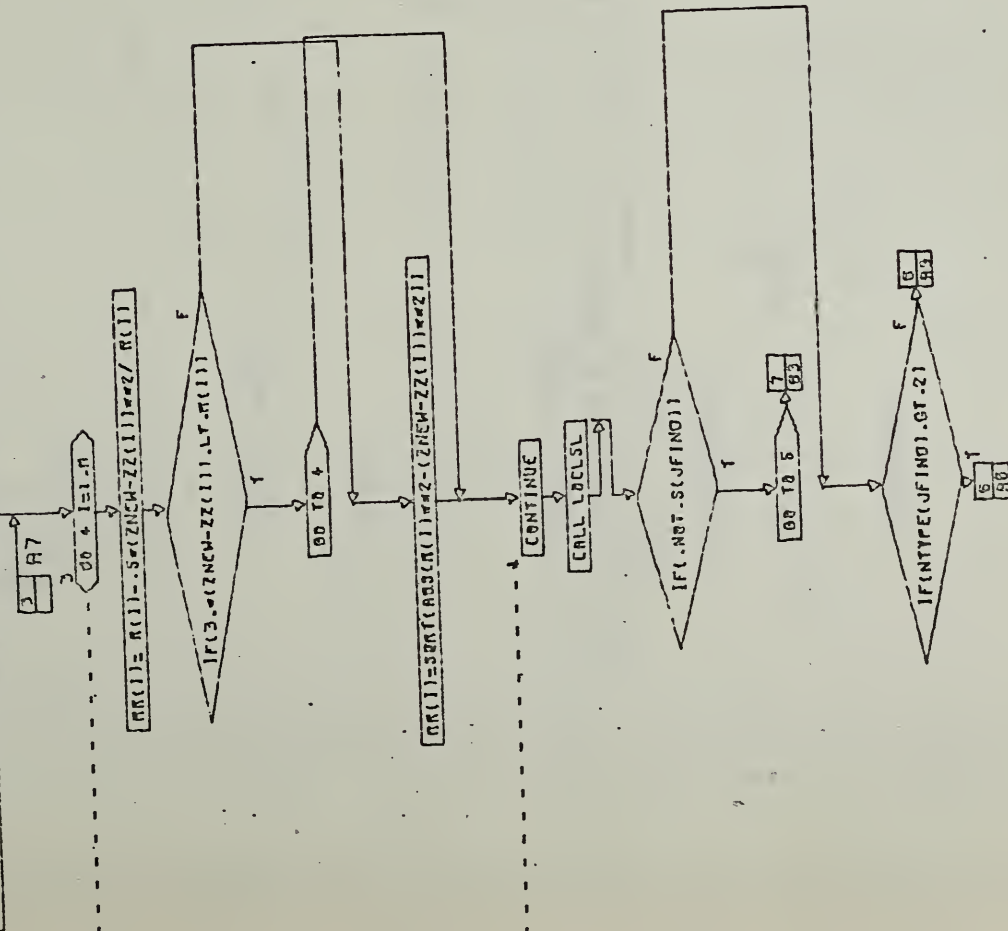
P0 4 OF 13

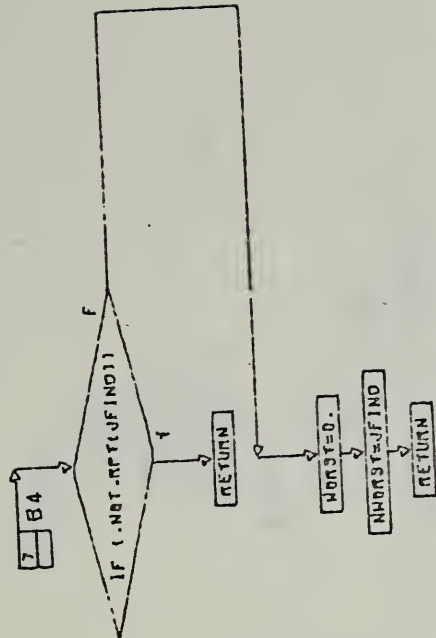


CONT. ON P0 4

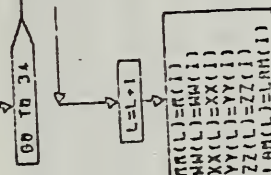
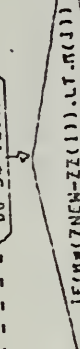
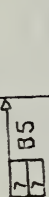
P0 3 OF 13

THE LAST ATTEMPT TO LOCATE THE LOCATEE WAS NOT SUCCESSFUL. SO AN ESTIMATE OF THE PRESENT POSITION MUST BE FOUND BY CALLING LSL. FIRST, A SLANT-HEIGHT REDUCTION MUST BE DONE (3-D DISTANCES ARE CONVERTED TO PLANE DISTANCES).



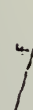
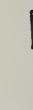


C FIVE LOCATORS HAVE BEEN FOUND. 30 LS300 IS CALLED.



CONT. ON P0 9

CO 9 OF 12

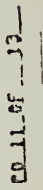


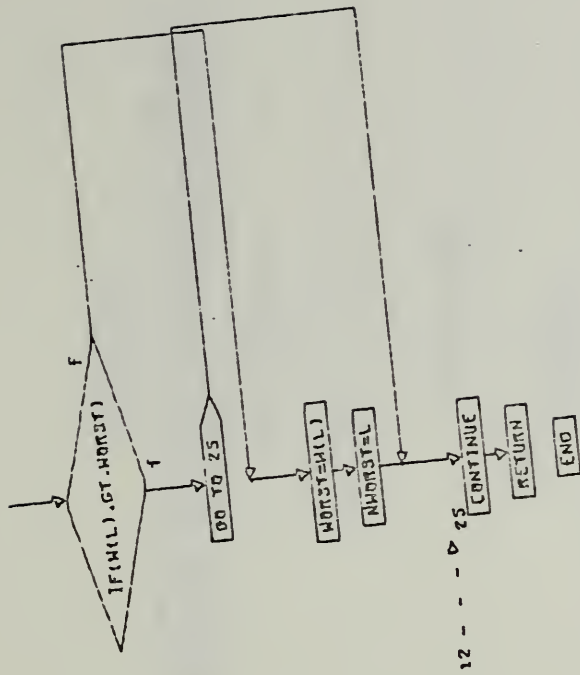
C THE LOCATEE CANNOT BE FOUND.

CONT. ON P0 8

CO 7 OF 13







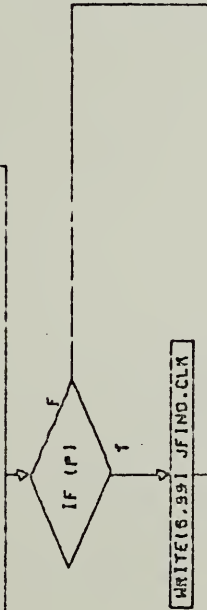
SUBROUTINE LQCLSL

C LEAST SQUARES LINEAR (LSL).
 C THIS ROUTINE MINIMIZES THE SUM OF THE SQUARED DISTANCES FROM THE
 C LOCATEE TO THE RADICAL AXIS LINES FORMED BY PAIRS OF CIRCLES.
 C EACH CIRCLE HAS A LOCATOR FOR ITS CENTER, AND THE REPORTED
 C DISTANCE TO THE LOCATEE FOR ITS RADIUS. SEE FIRST INTERIM REPORT
 C ON PROJECT WHENSH. PP. 10-13.
 C THIS IS THE TWO-DIMENSIONAL ROUTINE.

PARAMETER L1=225,L4=L1+1
 A(400),D(400),C(400),VRSQ(400)

COMMON/CO1/EP(L1-14),LPM(L4),MT(L4),JLOC(L1-4),H(L1),
 JFIND,CLM,XNEW,TNEW,FXD(L1),MPT(L1),NTYPE(L1)

COMMON/XYZ/M,L,XX(100),YY(100),ZZ(100),MM(100),N(100),P
 LOGICAL S,F
 DATA TOL/50./
 P=.FALSE.

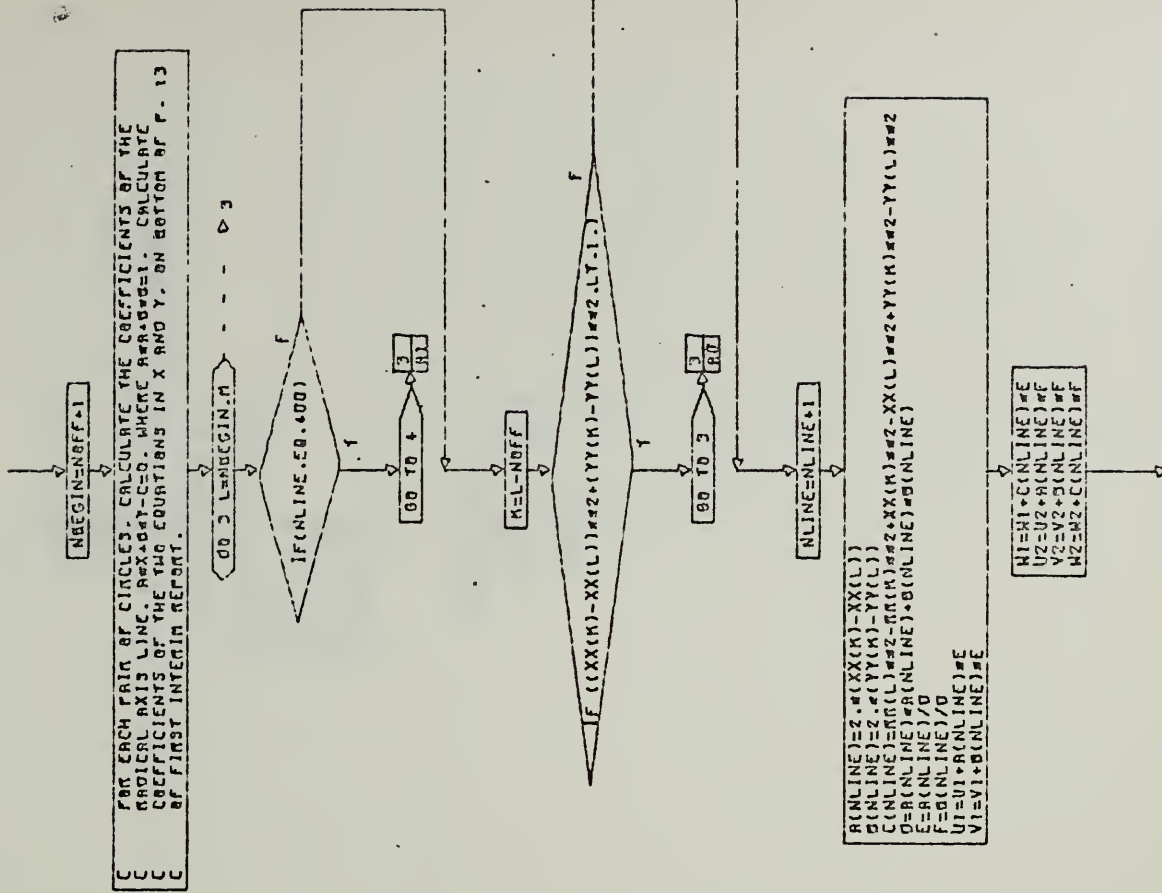


FORMAT(1) = 'LSL20: UNIT = 'IS.' CLOCK = 'F10.4'

S(JFIND)=.FALSE.
 NCLIM=4
 MCLIM=0
 NN=0-1

NLINE=0
 U1=0.
 U2=0.
 V1=0.
 V2=0.
 H1=0.
 H2=0.

D0 3 NOFF=1,NN



C FOR EACH PAIR OF CIRCLES, CALCULATE THE COEFFICIENTS OF THE
 C RADICAL AXIS LINE. A=XX(M)-XX(L). WHERE A=XX(M)-XX(L). CALCULATE
 C COEFFICIENTS OF THE TWO EQUATIONS IN X AND Y, ON BOTTOM OF P. 13
 C OF FIRST INTERIM REPORT.

D0 3 L=NOCIN,M

IF(NLINE.EQ.400)

00 TO 3

M=L-NOFF

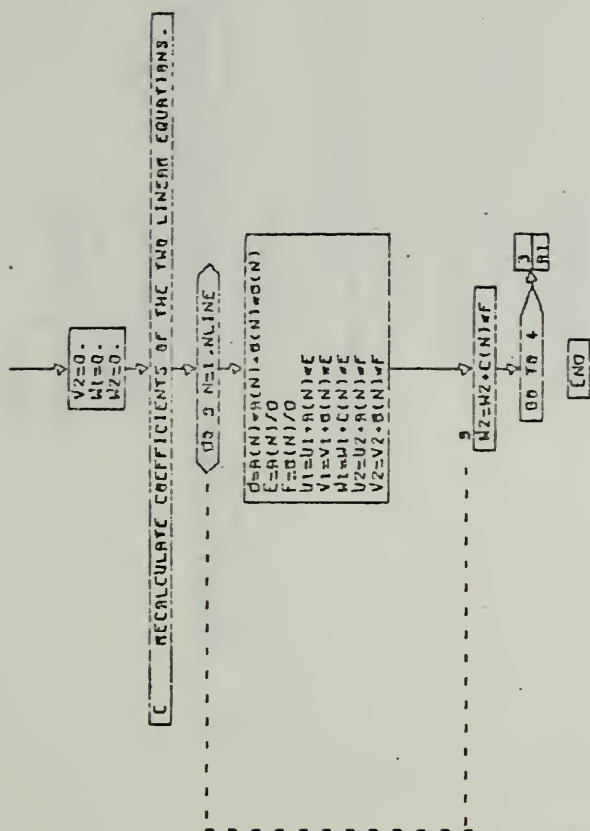
IF ((XX(M)-XX(L))**2+(YY(M)-YY(L))**2.LT.1.)

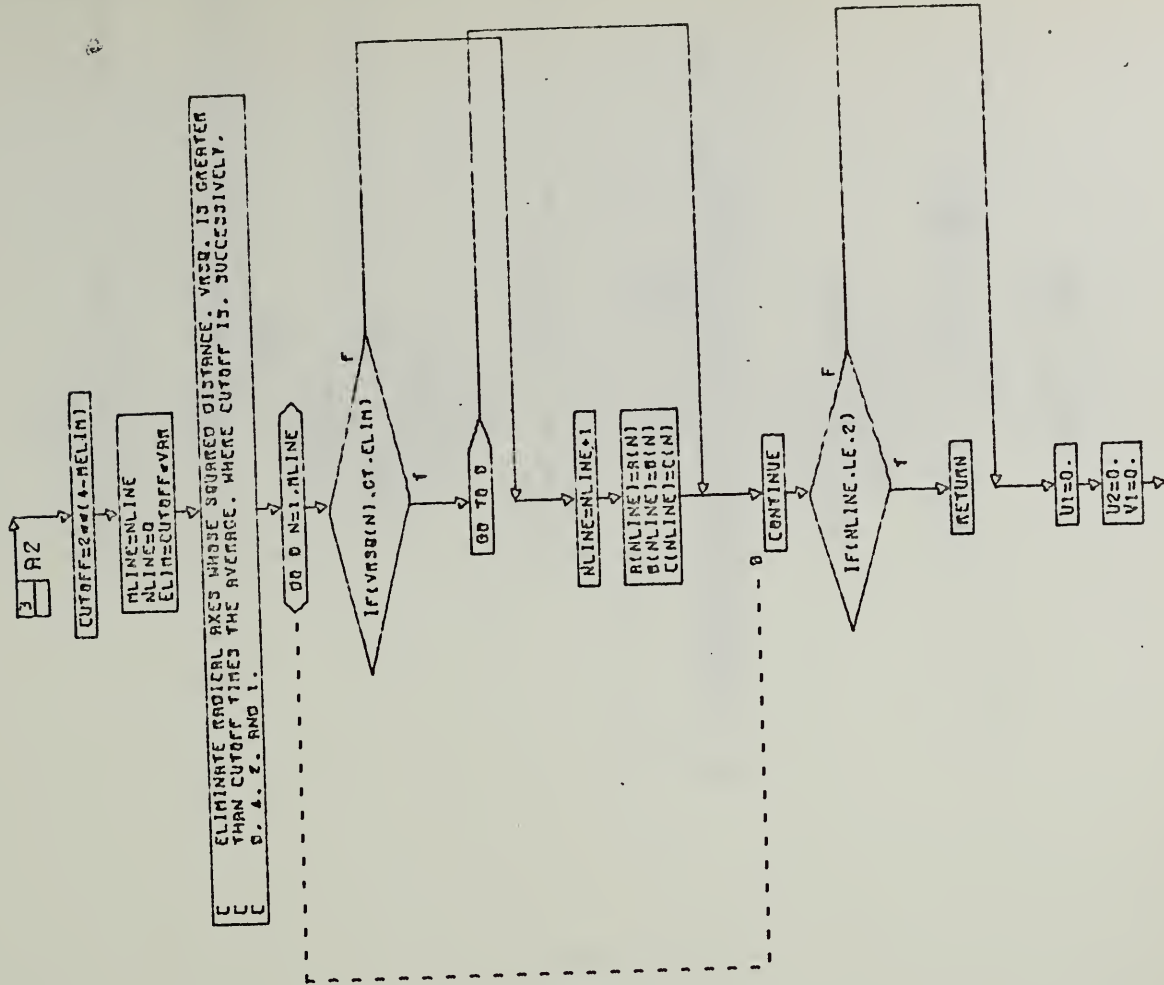
00 TO 3

NLINE=NLINE+1

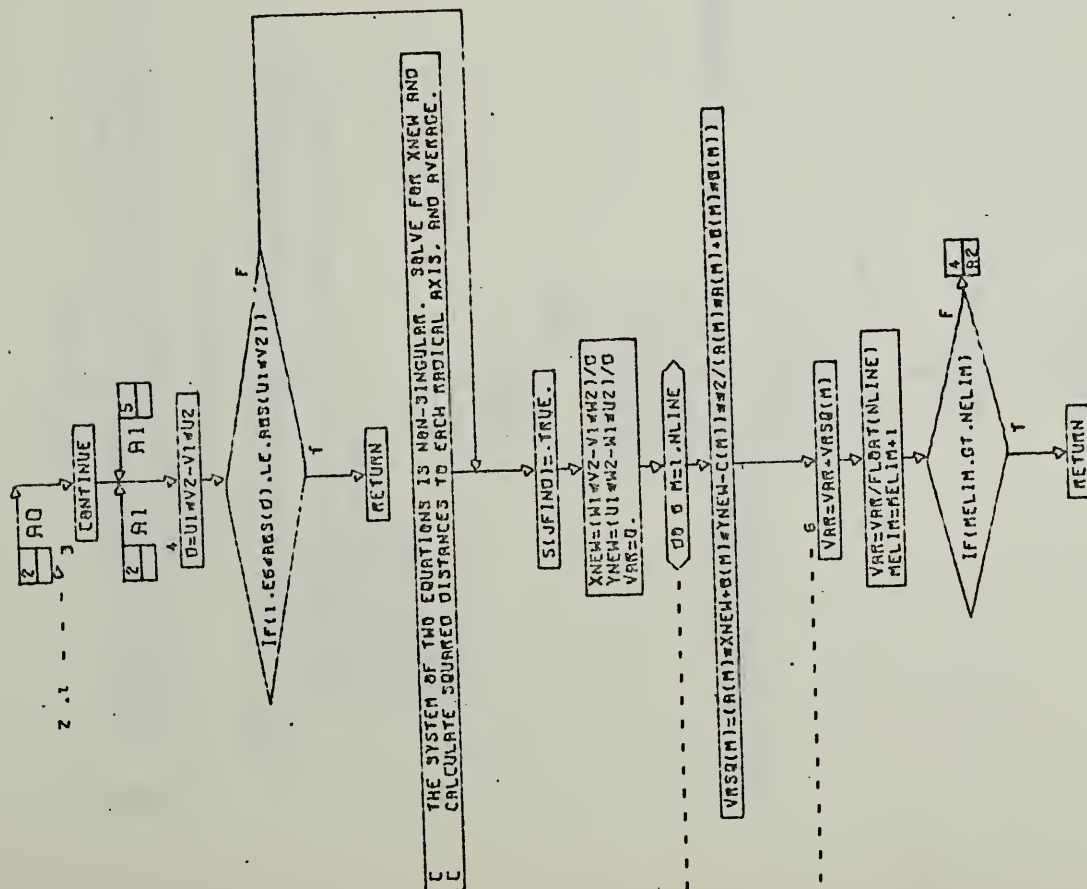
R(NLINE)=2.*(XX(M)-XX(L))
 G(NLINE)=2.*(YY(M)-YY(L))
 C(NLINE)=2.*(XX(L)**2-XX(M)**2-XX(L)**2+YY(L)**2-YY(M)**2)
 D=R(NLINE)*R(NLINE)+G(NLINE)*G(NLINE)
 E=D(NLINE)/D
 F=D(NLINE)/D
 U1=U1+R(NLINE)*E
 V1=V1+G(NLINE)*E

H1=H1+C(NLINE)*E
 U2=U2+D(NLINE)*E
 V2=V2+D(NLINE)*E
 H2=H2+C(NLINE)*E





CONT. ON PO 5



CONT. ON PO 4

PO 3 OF 5

PO 4 OF 5

SUBROUTINE LOCLES

C THIS IS THE TWO-DIMENSIONAL LEAST SQUARES SQUARED ROUTINE. IT
 C ATTEMPTS TO MINIMIZE THE FUNCTION ERVE BY FINDING A POINT WHERE
 C ITS DERIVATIVES F AND G, WITH RESPECT TO X AND Y ARE ZERO. TO DO
 C THIS, IT USES NEWTON-RAPHSON ITERATION.

PARAMETER LI=225, LA=LI+1

COMMON /C01/CP(LI,14),LAP(L4),MR(L4),S(L1),TLOC(L1,4),W(L1),
 UFINO,CLA,XNCH,YNCH,ZNCH,FXO(L1),GYO(L1),NTYPE(L1)

COMMON/MXZ/M,L,XX(DD),YY(DD),ZZ(DD),WN(DD),R(DD),F
 DIMENSION EE(DD)
 DATA NELIM,NMLOC,FARC/1.0,1/
 LOGICAL F,STOP
 DATA F//.FALSE./
 LR=0
 L=1

K=0
 NMLOC=0

DO 4 R=1

STOP=.FALSE.

OLDERVE=1.E30
 MELIM=0

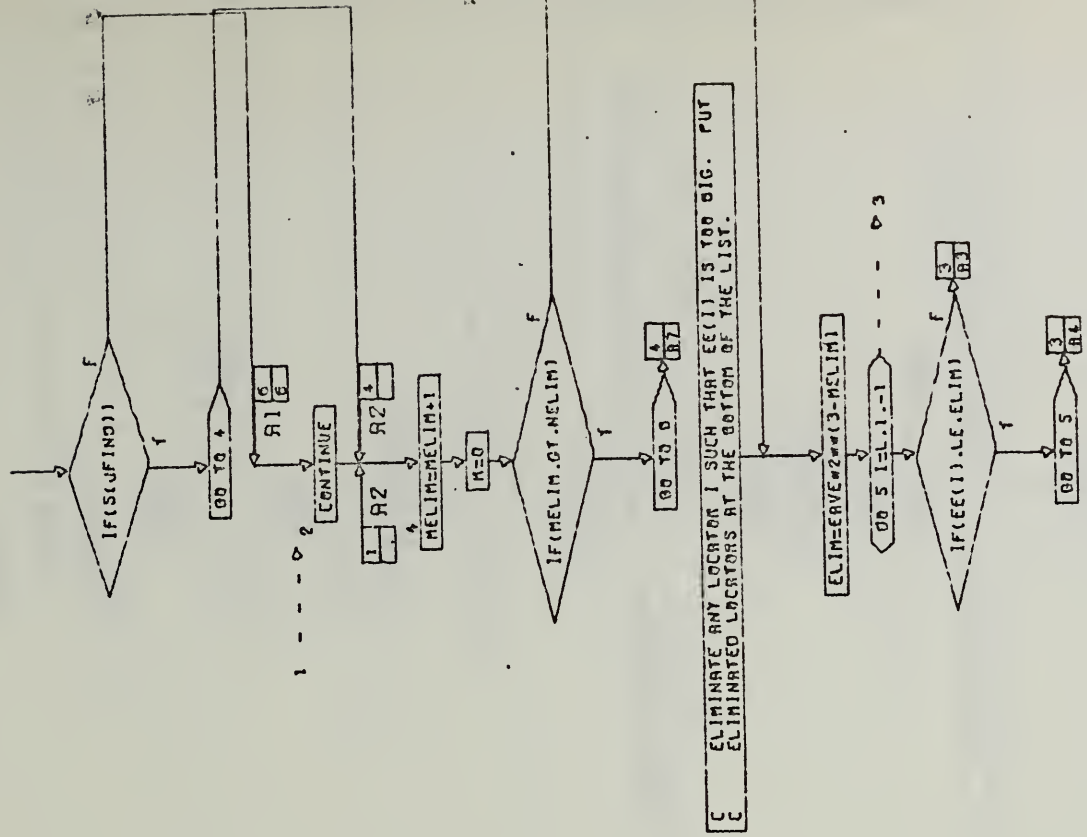
C DO UP TO FOUR NEWTON-RAPHSON ITERATIONS. STOPPING IF CONVERGENCE
 C OR DIVERGENCE OCCURS.

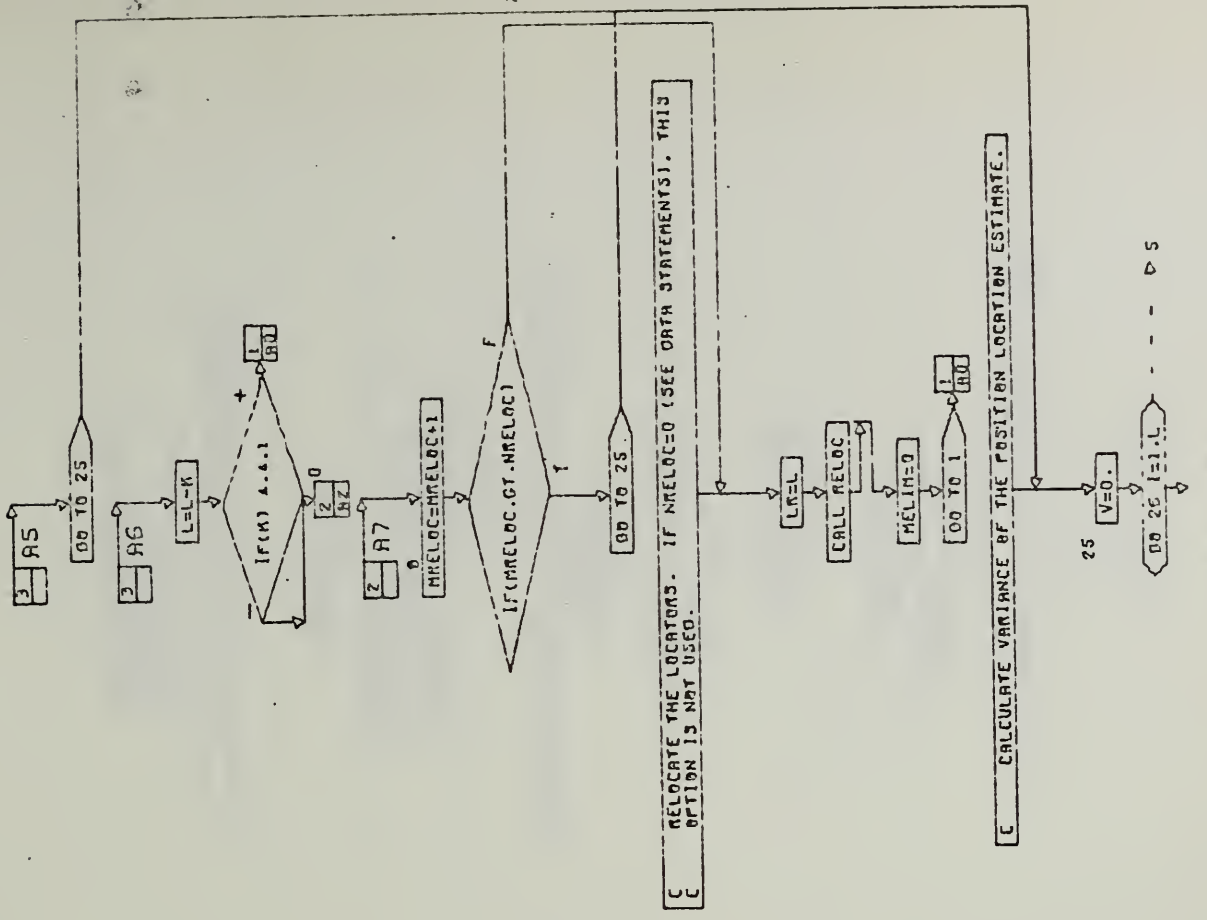
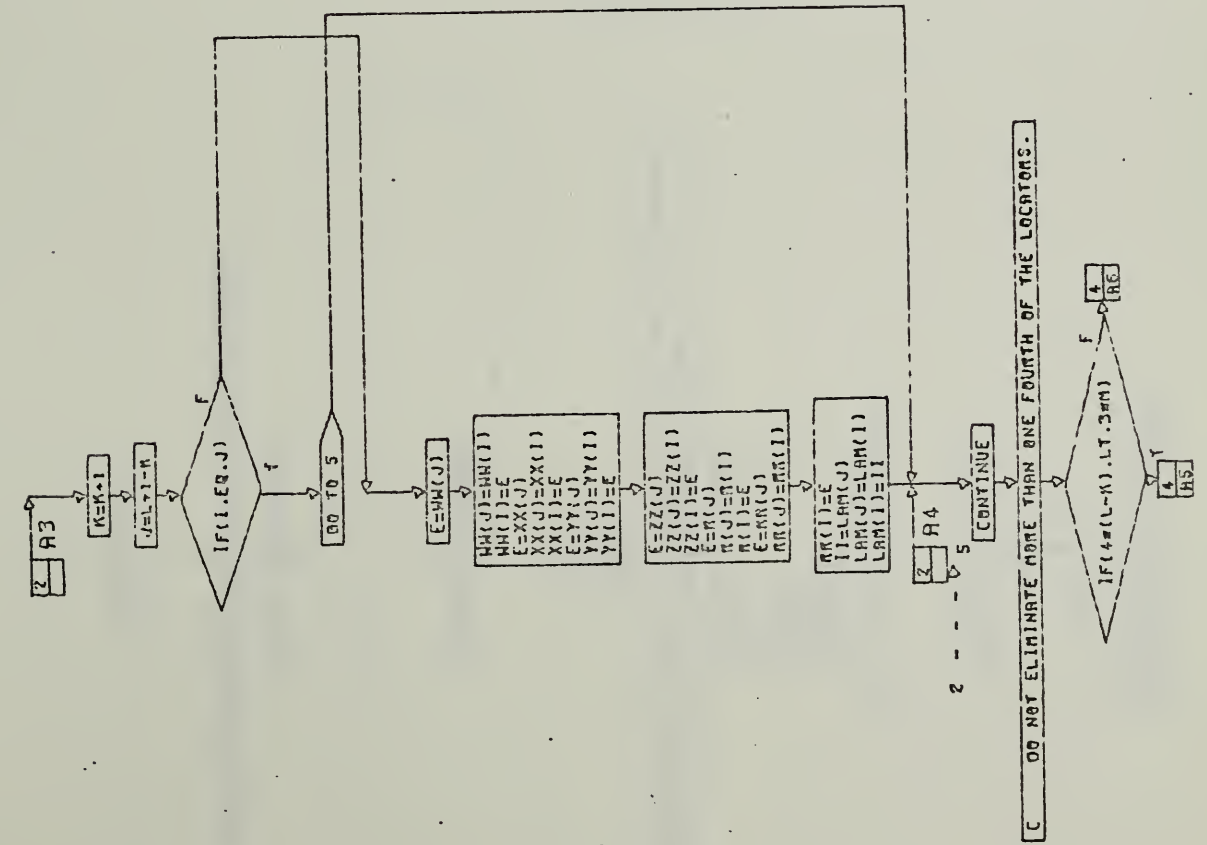
DO 2 I=0,3

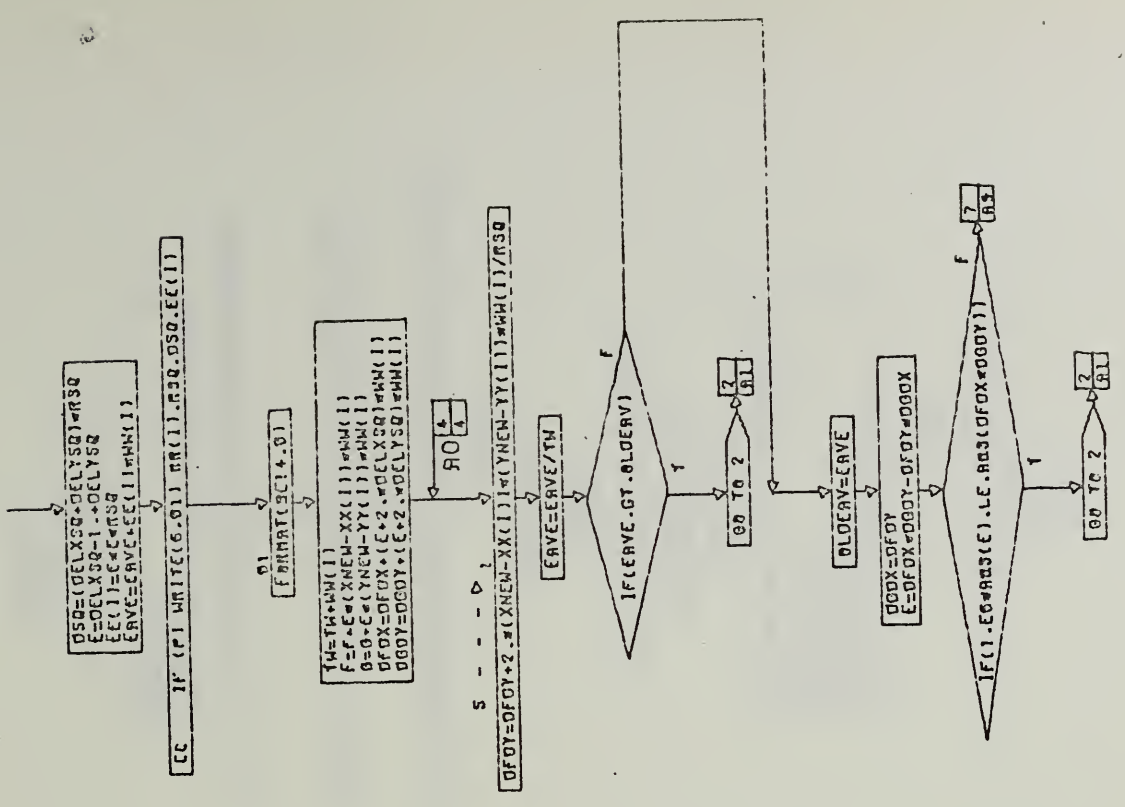
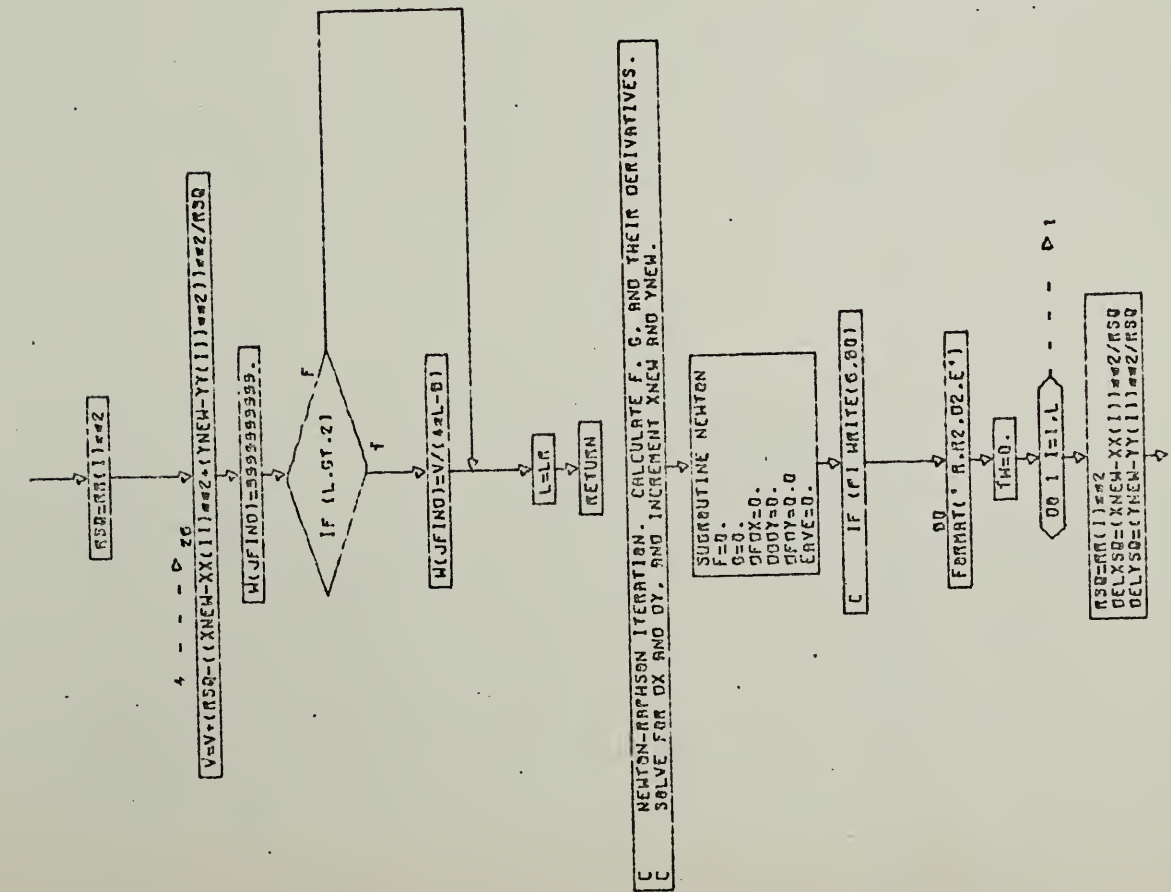
IF(STOP)

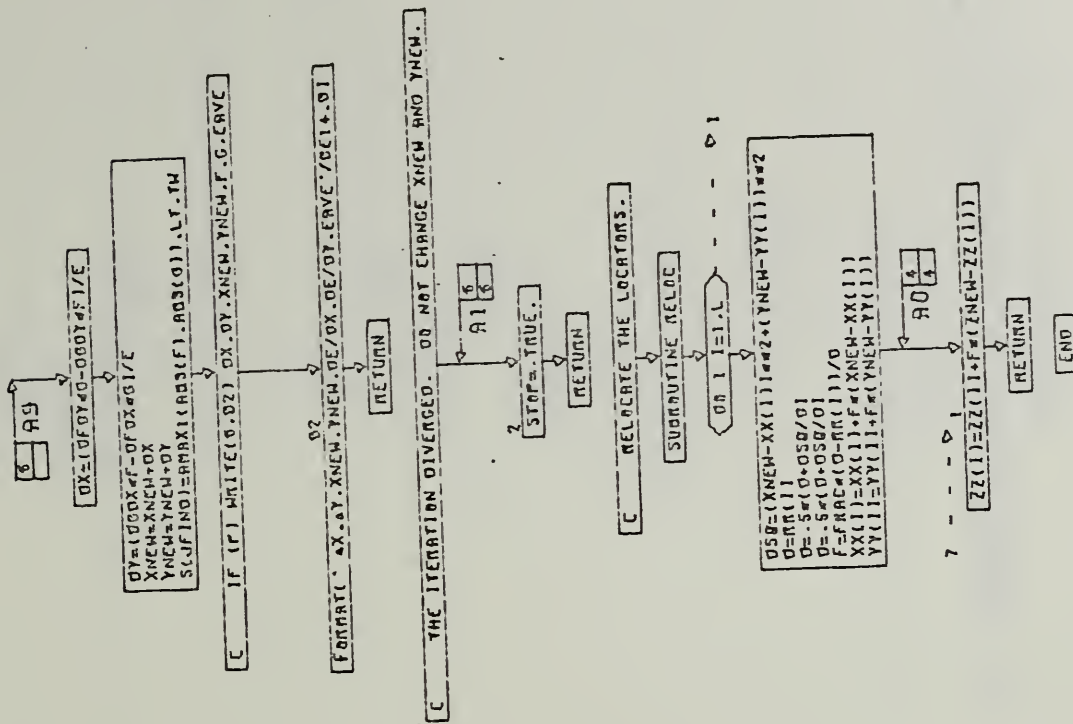
DO 4 R=1

CALL NEWTON









SUBROUTINE LSS30

C THIS SUBROUTINE IS THE 3-DIMENSIONAL LEAST SQUARES SQUARED METHOD.
C IT MINIMIZES A FUNCTION, CALLED BY TRYING TO GET ITS 3 DERIVATIVES
C (F, G, H RESPECTIVELY) WITH RESPECT TO X, Y, AND Z TO BE ZERO.
C NEWTON-RAPHSON ITERATION IS USED TO APPROACH THE ZEROES.

PARAMETER L1=225, L=1, L1=1

COMMON/COI/EP(L1,14),LAMB(L1,4),JCL(L1),JLBC(L1,4),JLCL(L1),
JFINO,CLK,XNCH,TNCH,FXO(L1),JOT(L1),NTIME(L1)

COMMON/XYZ/A-L,XX(L00),YY(L00),ZZ(L00),WW(L00),H(L00),F
DIMENSION CCL00
LOGICAL S,STOP,F
DATA MELIM,MRELBC,FHRC/1.0,0.2/

C IF (F) WRITE(6,99) JFINO,CLK

FORMAT 99 = LSS30: UNIT = 15, CLCK = F10.4)

L=0
L=M
M=0
MRELBC=0

90 1 9

1 STOP=.FALSE.

ACLM=0
ALOEAV=1.E30
CX=10000.
CY=10000.
EZ=10000.

C DO UP TO FOUR ITERATIONS. STOPPING IF CONVERGENCE OCCURS, OR IF
C THE FIRST ITERATION DIVERGES.

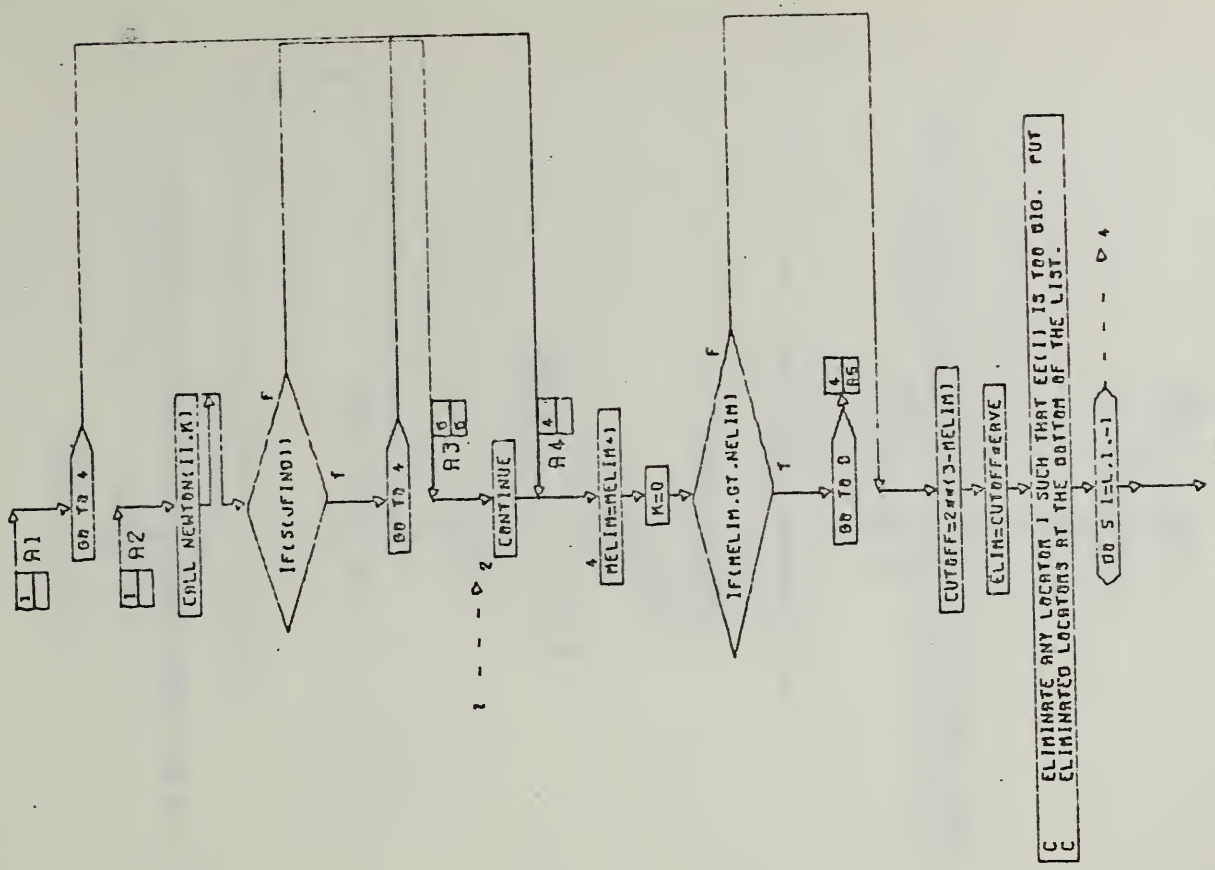
DO 2 11=0,3

IF(STOP) 1

2 11

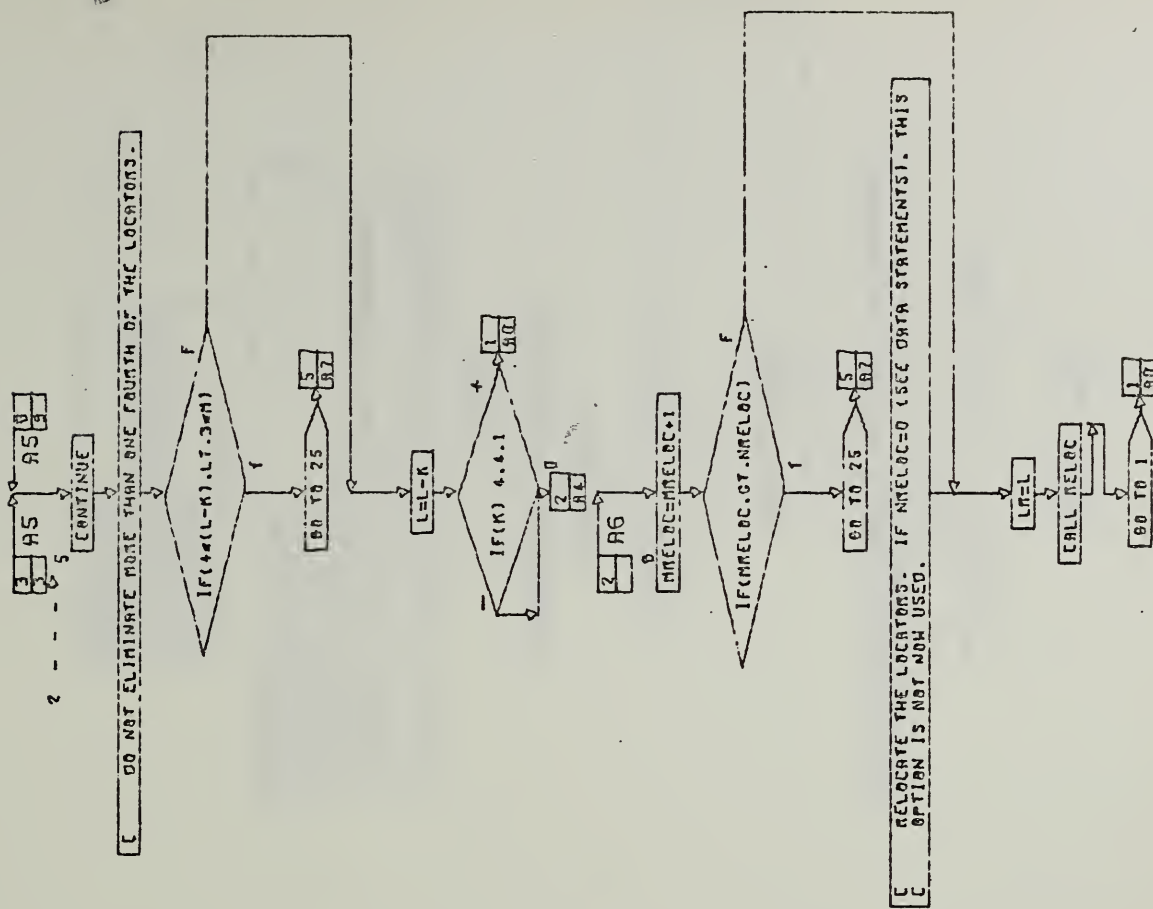
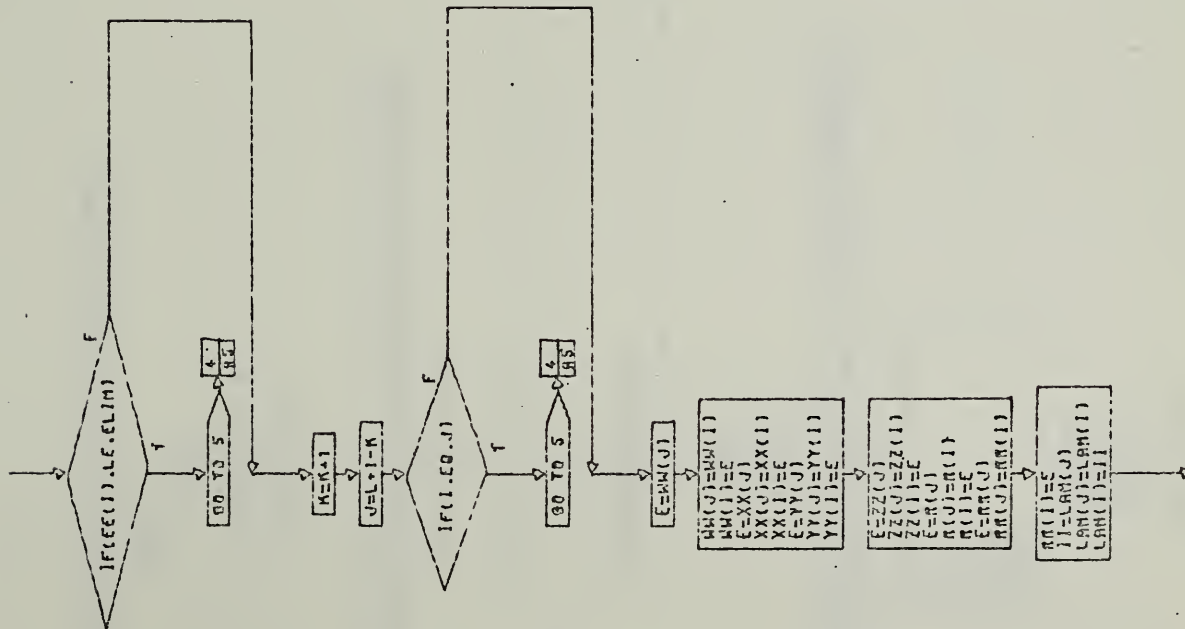
CONT. ON PG 2

CO.1 OF 10

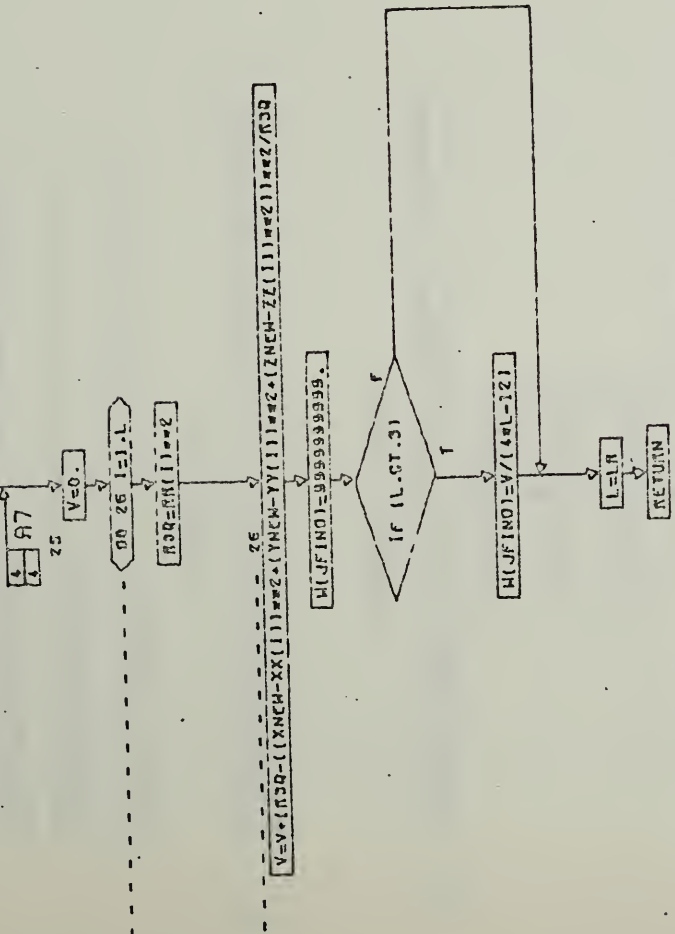


CONT. ON PG 3

CO.2 OF 10



C CALCULATE VARIANCE OF THE POSITION ESTIMATE.



C NEWTON-RAPHSON ITERATION. CALCULATE F, C, H AND THEIR DERIVATIVES
C AND FIND OX, OY, OZ BY SOLVING EQUATIONS (5) OF WORKING PAPER NO.
C 7 OF THE FINAL REPORT (MAY 1972).

SUBROUTINE NEWTON(I,N)
F=0.
C=0.
H=0.
OFOX=0.
OFOY=0.
OFOZ=0.
OGOT=0.

OG0Z=0.
OHOZ=0.
EAVE=0.
TH=0.

GO 1 I=1,L

CONT. ON P0 6

R30=RR(1)=2
R3QINV=1/R3Q
OCLX3Q=OCLX3Q*(XNCM-XX(1))=2
OCLY3Q=OCLY3Q*(YNCM-YY(1))=2
OCLZ3Q=OCLZ3Q*(ZNCM-ZZ(1))=2
E=OCLX3Q*OCLY3Q-1.-OCLZ3Q
EC(1)=E/E
EAVE=EAVE+EC(1)*HH(1)

TH=TH+HH(1)
F=F+C*(XNCM-XX(1))=HH(1)
C=C+C*(YNCM-YY(1))=HH(1)
H=H+C*(ZNCM-ZZ(1))=HH(1)
OFOX=OFOX+(E+2.*OCLX3Q)*HH(1)
OFOY=OFOY+2.*XNCM-XX(1)=(YNCM-YY(1))*R3QINV+HH(1)
OFOZ=OFOZ+2.*XNCM-XX(1)=(ZNCM-ZZ(1))*R3QINV+HH(1)
OGOT=OGOT+(E+2.*OCLY3Q)*HH(1)
OG0Z=OG0Z+2.*YNCM-YY(1)=(ZNCM-ZZ(1))*R3QINV+HH(1)

5 - - - ->
OHOZ=OHOZ+(E+2.*OCLZ3Q)*HH(1)

EAVE=EAVE/TH

IF(EAVE.GT.0LOERV)

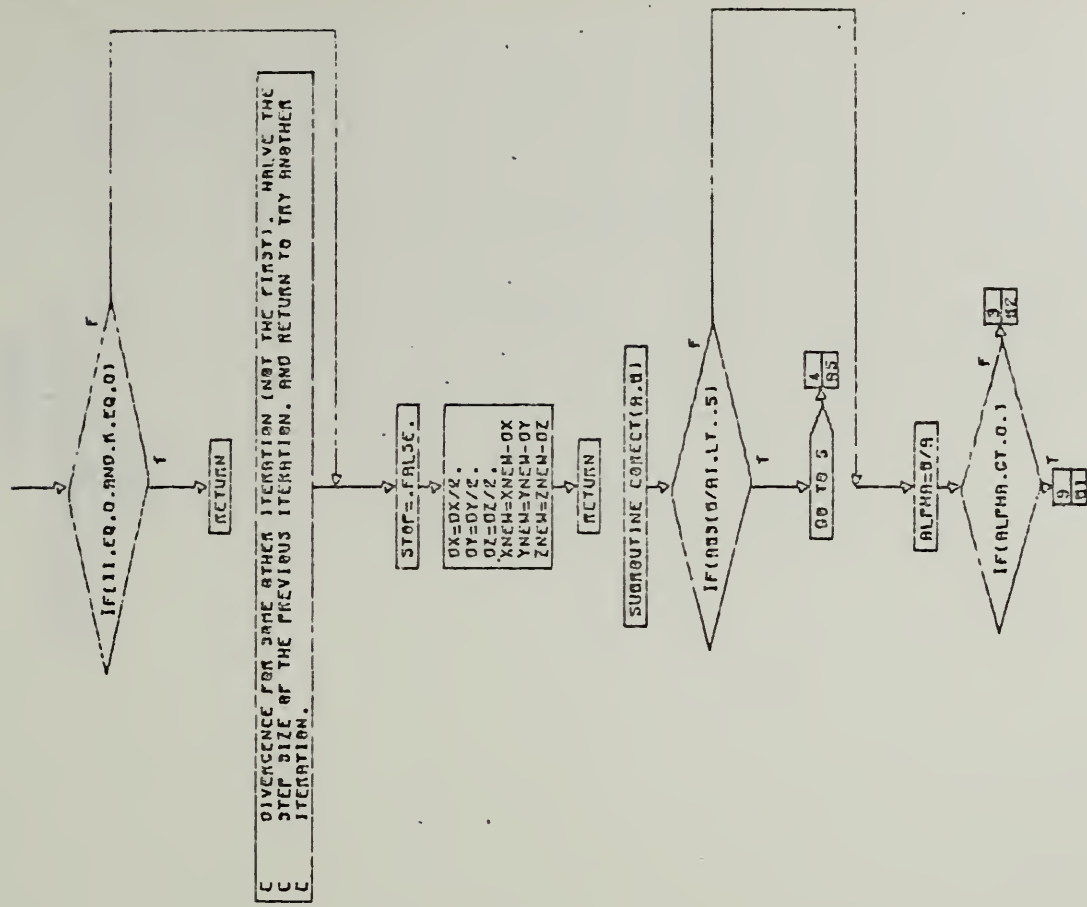
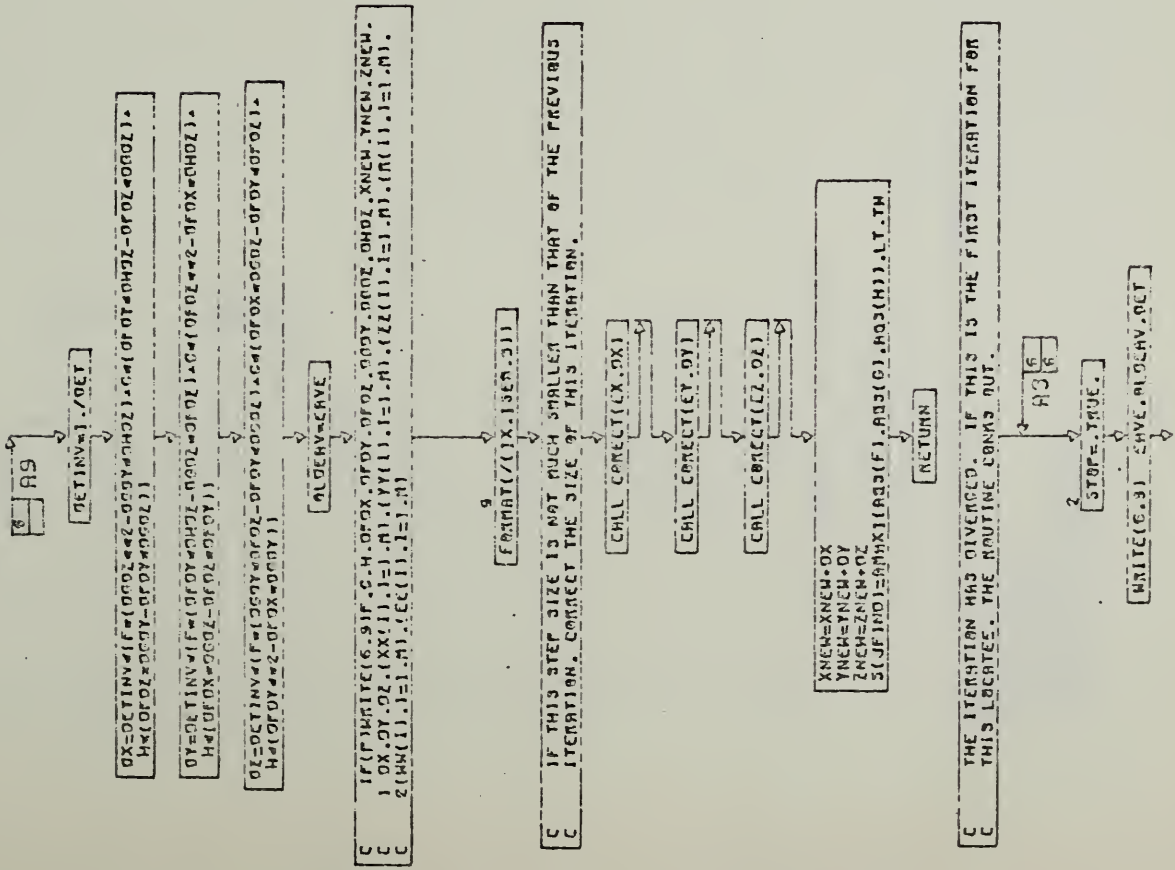
GO TO 2

DET=OFOX*(OFOY*OHOZ-OG0Z*2)+OFOY*(OG0Z=OFOZ-OFOY*OHOZ)
+OFOZ*(OFOY*OG0Z-OG0Y*OFOZ)

IF(R3Q(OET1.LT.1.-E-0))

GO TO 2

CONT. ON P0 7



PROJECT " WHERE " : WORKING PAPER NO. 8

LS, LSS, AND RELOCATION

James A. Lechner

April, 1972

ABSTRACT

In Working Paper No. 2 it was suggested that since the locators' positions are not known exactly, it might prove profitable to re-estimate these positions as an adjunct to estimating the position of the "locatee" unit. This paper presents an iterative method, along the lines proposed in Working Paper No. 2, for handling the computations of such a "relocation" process. The discussion is carried out in the context of the LS(least squares) and LSS(least squares, squared) position-location methods. As background, evidence is given which shows that these two criteria behave very similarly so long as range measurement errors remain small.

Note: Project working papers are informal documents prepared to facilitate discussion and communication; they may contain tentative or relatively unchecked material.

CONTENTS

0.	INTRODUCTION -----	1
1.	RELATION BETWEEN LS AND LSS APPROACHES -----	3
2.	THE RELOCATION PROCESS -----	7

0. INTRODUCTION

In this paper the term location-operation will be used for a process of attributing a position to one unit (the locatee) on the basis of its recorded distances from n other units (the locators) . These recorded distances are of course subject to measurement error.

Roughly speaking, the situation studied in Project "WHERE" is one in which the role of "locatee" varies over some totality of units, with many of these units in motion at any time. From this it is apparent that for any one location-operation, the positions of the locators (which may have been the locatees of previous operations) are not in general known exactly. This point was emphasized in Section 5 of Working Paper No. 2 [" An Iterative Approach to the Location-Operation" , A.J. Goldman, 10/71], where it was suggested that the location-operation be expanded to include re-estimation (i.e., relocation) of the locators' positions, based on the latest set of range measurements.

The present paper has as its main purpose the computational specification of such a relocation procedure. This is given in Section 2, which also discusses the respective merits of the LS (least squares) and LSS (least squares, squared) position-location methods in this context. Section 1 presents some relevant background information concerning the relationship between these two algorithms.

We conclude this Introduction by introducing some necessary mathematical notation:

O = (arbitrary) origin of coordinate system ,

X = position to be attributed to locatee,

x = vector OX

A_i = position to be attributed to i -th observer
($i = 1, 2, \dots, n$) ,

a_i = vector OA_i ,

α_i = initial estimate of a_i ,

r_i = $\|x - a_i\|$ = estimated distance between locatee
and i -th observer,

ρ_i = measured value of this distance .

If the distinction between a_i and α_i is dropped (i.e., the initial estimates of the locators' position are accepted without revision), then we have the "pure" location-operation, which can be expressed as the problem of making a "good" choice of x , given the ρ_i and a_i . The LS method expresses this problem as that of minimizing the quadratic "penalty function"

$$f(x) = \sum_i w_i' (r_i - \rho_i)^2 \quad (0.1)$$

where the w_i' are positive "weights" and x enters via the r_i 's ;
the LSS approach involves the minimization of

$$g(x) = \sum_i w_i (r_i^2 - \rho_i^2)^2 , \quad (0.2)$$

with "weights" w_i . The modification of these formulations, to encompass relocation, will be taken up in Section 2 .

1. RELATION BETWEEN LS AND LSS APPROACHES

In this section, as background for the discussion of relocation in Section 2, we discuss the relation between the LS and LSS approaches to the "pure" location-operation. Three pieces of evidence will be given to show that if the "weights" in these two criteria --- the w'_i of (0.1) and the w_i of (0.2) --- are properly related, then the two methods will not differ significantly in accuracy so long as the tracking process is under control. (The last phrase means that the units' positions remain well enough known that the discrepancies between measured ranges ρ_i and calculated ranges r_i are small relative to the magnitudes of the ranges themselves.)

The first piece of evidence begins with the observation that the "under control" assumption, $r_i/\rho_i \approx 1$, yields

$$\begin{aligned}(r_i^2 - \rho_i^2)^2 &= (r_i + \rho_i)^2 (r_i - \rho_i)^2 \\ &= (1 + r_i/\rho_i)^2 \rho_i^2 (r_i - \rho_i)^2 \\ &\approx 4\rho_i^2 (r_i - \rho_i)^2,\end{aligned}$$

or equivalently

$$(1/4\rho_i^2)(r_i^2 - \rho_i^2)^2 \approx (r_i - \rho_i)^2.$$

Thus, if we set $w_i = (1/4\rho_i^2) w'_i$, then comparison of (0.1) with (0.2) shows that $g(x) \approx f(x)$; equivalently, if we set

$$w_i = w'_i / \rho_i^2 \tag{1.1}$$

then comparison of (0.1) yields $g(x) \approx 4f(x)$, so that minimization of $f(x)$ --- i.e., the LS approach --- is "approximately equivalent" to the LSS approach (minimization of $g(x)$) .

Equation (1.1) gives the appropriate relation between the two sets of weights. As yet, no rationale has been found for assigning to the w_i' any value other than unity. In fact, since the distribution of errors in (unrejected) ranges seems (by best engineering judgement at this time) to be Gaussian, with range-independent variance and zero mean (except for a small percentage of ranges biased by about 8m), maximum likelihood estimation would strongly recommend⁽¹⁾ use of LS with $w_i' = 1$ (together with some technique for removing "outliers"), which by the above is approximately equivalent to use of LSS with

$$w_i = 1/\rho_i^2 . \quad (1.2)$$

This choice of w_i' and w_i will be used throughout the rest of this paper.

The preceding analysis shows that the penalty functions for LS and LSS are approximately equal (when measured and calculated ranges are close together), but one would like some direct assurance that the position-estimates obtained by minimizing these two functions are also nearly equal. This is provided by our second piece of evidence, an analysis for a simple 1-dimensional configuration in which the locators are situated at the points with coordinates R and $(-R)$ with R large, while the locatee is actually situated at the origin. The LS approach calls for the minimization of

$$\begin{aligned} f(x) &= [(R-x) - \rho_1]^2 + [(R+x) - \rho_2]^2 \\ &= 2\{x^2 + (\rho_1 - \rho_2)x\} + \text{const.}, \end{aligned}$$

(1) See Section 1 of Working Paper No. 2 .

which is readily found to be given by

$$x = (\rho_2 - \rho_1)/2 \quad . \quad (\text{LS solution}) \quad (1.3)$$

The LSS approach calls for the minimization of

$$\begin{aligned} g(x) &= \rho_1^{-2} [(R-x)^2 - \rho_1^2]^2 + \rho_2^{-2} [(R+x)^2 - \rho_2^2]^2 \\ &= \rho_1^{-2} (R-x)^4 + \rho_2^{-2} (R+x)^4 - 4x^2 + \text{const.} ; \end{aligned}$$

setting $dg/dx = 0$ leads to the cubic equation

$$(\rho_1^2 + \rho_2^2)x^3 + 3R(\rho_1^2 - \rho_2^2)x^2 + [3R^2(\rho_1^2 + \rho_2^2) - 2\rho_1^2\rho_2^2]x + R^3(\rho_1^2 - \rho_2^2) = 0 \quad . \quad (1.4)$$

Let E denote a measure of maximum range-error. In the worst-case scenario described by $\rho_1 = R-E$ and $\rho_2 = R+E$, the LS and LSS approaches yield exactly the same answer, both (1.3) and (1.4) giving $x = E$. In the alternative scenario described by $\rho_1 = R$ and $\rho_2 = R+E$, the respective solutions are

$$x_{\text{LS}} = \frac{1}{2}E \quad , \quad x_{\text{LSS}} = \frac{1}{2}E \left[1 - \frac{3}{8}(E/R)^2 \right] \quad (1.5)$$

where the expression for x_{LSS} omits higher-order terms in the small quantity E/R ; if for example R is at least 1 km and E is at most 20m, then the two position-estimates differ by at most 0.0015 m.

Our third piece of evidence is empirical in nature. For each of two distributions of range-measurement errors ("small" and "large", respectively), 72 trial 2-dimensional location-operations were carried out using both LS and LSS . Each of 18 units served as locatee 4 times (for each error distribution and each method), with the other 17 units serving as locators. For the "small" case, the locations were as much as 3.32m in error, but the LS and LSS results differed from each other by no more than 0.0004m ; for the "large" case, despite location errors up to 31.9m, the two methods yielded locations no more than 0.045m apart.

2. THE RELOCATION PROCESS

When the location-operation is enlarged to include relocation (of the locators), it can be formulated as making a "good" choice of x and the a_i , given the ρ_i and α_i . The approach given in Section 4 of Working Paper No. 2 is based on two notions, one of which we have found it is useful to modify in a manner described below.

The first notion is that the choices of x and a_i 's should be made so as to minimize a penalty function which generalizes those given earlier [see (0.1) and (0.2)] for "pure" location. Specifically, this function is

$$f(x; a_1, \dots, a_n) = \sum_i w_i' (r_i - \rho_i)^2 + \sum_i W_i \|a_i - \alpha_i\|^2 \quad (4.1a)$$

for the LS approach, and

$$g(x; a_1, \dots, a_n) = \sum_i w_i (r_i^2 - \rho_i^2)^2 + \sum_i W_i \|a_i - \alpha_i\|^2 \quad (4.1b)$$

for the LSS approach ; the W_i are positive numerical weights.

The second notion is to accomplish this minimization by a process which alternates between location steps (in which the a_i are held fixed and an appropriate x is chosen) and relocation steps (in which x is held fixed and appropriate a_i 's are chosen. This process can be formalized as follows:

INITIALIZATION: Set $k = 1$, and $a_i^{(1)} = \alpha_i$ (all i).

STEP 1 (location, k -th pass) : Given the current estimates $a_i^{(k)}$ of the locator positions, choose $x = x^{(k)}$ to minimize the first sum in (4.1), i.e. the part of (4.1) which depends on x .

STEP 2 (relocation, k-th pass) : For each i , keep x fixed at $x^{(k)}$ and choose a_i to minimize the sum of those terms in (4.1) which depend on a_i . Increase k by 1, and set each $a_i^{(k)}$ equal to the a_i just found. Return to Step 1.

For Step 1, the material in Section 1 shows that there is little reason to prefer one of LS or LSS over the other so far as location-errors are concerned. We therefore select LSS, since it seems likely to converge more quickly and requires fewer time-consuming calculations per iteration. Thus Step 1 employs (4.1b) rather than (4.1a), and so according to the description above, the treatment of the i -th locator in the k -th relocation step would require choosing a_i to minimize

$$w_i \{ \|x^{(k)} - a_i\|^2 - \rho_i^2 \}^2 + w_i \|a_i - \alpha_i\|^2. \quad (4.2)$$

For Step 2, however, it is more convenient to minimize instead the sum of corresponding summands from (4.1a), namely

$$h(a_i) = w_i \{ \|x^{(k)} - a_i\| - \rho_i \}^2 + w_i \|a_i - \alpha_i\|^2. \quad (4.3)$$

The over-all method becomes a kind of LS-LSS hybrid ; Steps 1 and 2 in effect involve different penalty functions, so that the "first notion" described above has indeed been modified.

In Working Paper No. 2 it was envisaged that the alternating process would be iterated until it had "settled down" satisfactorily. This idea, too, has been modified after further reflection. The relocation of the i -th locator is based mainly upon a single new datum, namely ρ_i ; it seems

unwise to put too much reliance or to base too much calculation on this one datum (which is subject to measurement-error). Besides, from the viewpoint of reducing total computation time it is desirable to limit the number of iterations. Thus a sensible compromise seems to be: locate the locatee (Step 1); relocate the locators (Step 2) ; then locate the locatee again (second pass through Step 1), and stop. This truncated version of the full process is assumed below.

We turn now to the execution of Step 2. With superscripts (k) and subscripts (i) dropped for simplicity, the function (4.3) to be minimized can be written

$$h(a) = w' \{ \|x-a\| - \rho \}^2 + W \|a-\alpha\|^2 . \quad (4.4)$$

Consider any trial position a' of a . It lies on a sphere of radius $\|x-a'\|$ centered at x . Suppose a' is replaced by the point of that same sphere which is closest to α ; then the first summand in (4.4) is unchanged, while the second summand is reduced. Hence the minimum can only occur for such a "closest point". Simple geometric considerations show that such a point lies either on the line segment $[x, \alpha]$ or on its extension through α . Thus, for some scalar $s \geq 0$, we have

$$a = x + s(\alpha - x) ,$$

leading to

$$x - a = s(x - \alpha) , \quad a - \alpha = (1-s)(x - \alpha) .$$

Substitution of these results into (4.4) converts $h(a)$ into a function of s ; with the abbreviation

$$r = \|x - \alpha\| , \quad (4.5)$$

this function reads

$$\begin{aligned}
 H(s) &= w' \{sr - \rho\}^2 + Wr^2(1-s)^2 \\
 &= (w' + W)r^2 s^2 - 2r(w'\rho + Wr)s + (w'\rho^2 + Wr^2),
 \end{aligned}
 \tag{4.6}$$

and is minimized by setting

$$s = (w'\rho + Wr) / (w' + W)r . \tag{4.7}$$

(The availability of such a simple explicit solution to the minimization problem justifies our earlier assertion that use of (4.4) rather than (4.2) is more convenient.) If $\rho > r$ then $s > 1$, signifying that a lies on the extension of segment $[x, \alpha]$ through α (i.e., relocation has moved the locator farther from the locatee); if $\rho < r$, then $s < 1$ and a lies on the segment (so that the locator has been moved closer to the locatee). These results, which do not depend on w and W , seem quite consistent with what one would expect intuitively.

The only remaining question is the selection of numerical values for the weights W_i . We have at present no definitive answer to this problem. There are two distinct considerations to be taken into account:

(a) If we consider the current location (and relocation) operation in isolation, then the weight W_i should depend primarily on our confidence in the accuracy of the initial estimate α_i of the i -th locator's position (relative to the accuracies of the other α_j 's, and also of the range measurements). In an idealized Gaussian-error case, for example, maximum-likelihood estimation would dictate weights inversely proportional to the respective variances of the associated random errors. For our practical situation, how can we quantify the degree of confidence merited by the initial estimate α_i ?

It may well prove possible to concoct an appropriate measure based on the sizes of the discrepancies $|r_j - \rho_j|$ obtained during the last previous location-operation in which the i -th of the current locators figured as locatee---probably modified by some "decay factor" to take account of how much time has elapsed since this earlier operation.

(b) However, one needs to consider the "current" operation in the light of the need to maintain stability of the entire location process; the selection (or least the effect) of the W_i must reflect not only the one-shot independent-measurement case as in (a), but also the sequential correlated-error many-locator nature of the real problem. If a relocation (based mainly on a single and possible "bad" range measurement ρ_i) is allowed to be too large, its influence would persist through a number of operations until the unit in question again took its turn as locatee, and so such instabilities in estimation of positions could result in large errors.

The following tactic may prove an acceptable compromise. Weights W_i will be selected according to the considerations in (a) above. The resultant solutions (4.7) of the n minimization problems for relocation (1 problem for each of the n locators) will dictate moving the estimated position of the i -th locator a certain distance (from α_i) in a certain direction along the line through $x^{(k)}$ and α_i . To "temper" this solution in accordance with the considerations in (b), we will "damp" that distance by a multiplicative factor (say, between 0.1 and 0.6) before actually revising the estimate of the locator's position. Numerical experimentation will be required to test the performance of this approach and to determine what values of the adjustment factor are preferable.

PROJECT "WHERE": WORKING PAPER NO. 9

THE MNPLS SIMULATION ("WHERE SIM")

R.H.F. Jackson

May, 1972

Abstract: This paper documents the present version of the MNPLS simulation. The documentation is presented at three levels: the "executive" level (an overview), the "user" level (description of input and output), and the "analyst" level (detailed discussion of the logic flow). It focuses on the concepts indigenous to such a simulation study, namely: movement of units, range computation (including the occurrence of measurement error), and position estimation. A dictionary of variables used in the program, as well as a listing and a detailed flowchart of the program, are also included.

Note: Project working papers are informal documents prepared to facilitate discussion and communication; they may contain tentative or relatively unchecked material.

TABLE OF CONTENTS

	Page No.
1. Introduction	
2. Boxes 2 and 3: The Movement Modules	
3. Box 4: The Range Computation	
4. Boxes 5, 6, and 7: The Location Attempt, Statistics Accumulation and "End of Cycle" Test	
5. Boxes 8 and 9: The "Intervisibility Table Update"	
6. Boxes 10 and 11: The Production of Intermediate Output	
7. Box 12: Update the Stationary Unit List	
8. Boxes 13 and 14: Produce the X-Y Plots	
9. Box 15: The End of Simulation	
10. The Input to the Simulation	
11. The Output from the Simulation	
Appendix A: A Dictionary of Variables Used in the Simulation Programs	
Appendix B: Listing of the Programs	
Appendix C: Flowcharts of the Main Programs	

1. INTRODUCTION

WHERSM^(*) is a computer code that simulates the operation of a Micro-Navigation Position Location System (MNPLS). It provides a "real life" framework within which the propagation over time of position location errors, resulting from the operation of such a system, can be determined. It further provides a facility for monitoring the operation of such a system under a variety of movement scenarios, and using a variety of algorithms for accomplishing the location operation.

This paper documents the present version of the WHERSM code at three levels: the "executive" level, the "user's" level, and the "analyst's" level. This introductory section, which serves as the executive level of documentation, is aimed at the person who is interested only in an overview of what WHERSM is and what it does. The user's level of documentation is meant for one who is concerned with making a computer run with the code as it now stands; i.e., with no major program modifications. Sections 10 and 11 provide documentation of all the inputs and outputs -- the information needed to make a run. The remainder of the paper, Sections 2 through 9 and appendices A, B, and C, is designed with the analyst in mind. By an "analyst" is meant one who needs not only to run the program, but also perhaps to modify it to handle somewhat different situations. This paper should include enough information to start him on his way.

Due to the nature of the MNPLS under study, WHERSM is a "fixed time increment" simulation; i.e., simulated time is advanced by a constant increment, Δt , rather than by variable increments dictated by the occurrence of events, as would be the case in the type of simulation classified as "next event".

(*) Full name: WHERESIM. Compressed because of computer restrictions to 6 characters per name.

The program is written in FORTRAN V, but with very little conversion it can be compiled with a FORTRAN IV compiler. It is liberally annotated with comments, and although the code documented here employs program overlays, parallel processing, and drum usage, every attempt has been made to keep machine dependence minimized. Furthermore, the WHERSM code is modular in nature, so that the logic flow is kept simple and easy to understand, and so that different concepts in deployment, movement, intervisibility, position location method, etc., can with a minimum of reprogramming be implemented and tested. This modularity also allows extreme flexibility in the type and quantity of inputs required to make a run.

The basic ideas involved in an implementation of the MNPLS are that there are n units in the field, any of which at each moment might (in the most general case) be either moving or fixed. Every Δt seconds, a master computer receives information in the form of ranges, subject to measurement error, from one of these units to some or all of the others. The master computer must then, using position location algorithms, estimate the position of that unit and also prepare to receive the set of ranges for the next unit in sequence. The elements of this description that are relevant to a simulation study of an MNPLS are: movement, range computation (including the occurrence of measurement error), and position estimation.

In particular, the simulation must have some facility for determining the true positions of all the units at all relevant times. It is easy to provide a movement scenario in terms of movements in the xy-plane (longitude-latitude). A starting x and y , an azimuth of movement, and a speed will suffice for this^(*). It is the z coordinate, the height above sea level, that causes difficulty.

(*) As will be explained in Section 2, the movements are assumed to be piecewise linear, so that in fact a set of azimuths, speeds, and change times are required.

There are at least two methods of providing the z coordinate in a simulation. One involves a continuous function which yields a value of z for any values of x and y ; the other is by means of a discrete function which specifies values of z for particular values of x and y . In the discrete case, values of x and y other than the prescribed ones are assumed to coincide with the nearest available grid point, whose specified height is then used^(*). Both of these techniques are explained in detail in section 2.

The next item with which the simulation of the MNPLS must be concerned is the computation of ranges. Of course, given the ability to compute the true positions of all units at any instant of time, the true ranges can easily be calculated, since they are simply Euclidean distances. But, in the "real life" situation, true ranges are never reported to the master computer due to measurement errors intrinsic to the particular method of measuring those ranges. This phenomenon is easily simulated once a probability distribution of measurement errors is specified, since the simulation program can simply compute pseudo-random numbers which satisfy such a distribution law.

Another point relevant to range computation is that a simulation must be capable of determining whether a range should be computed; i.e., whether, for a specified pair of units at a specific moment, a radio line of sight (LOS) exists. There are many ways^(**) of providing a simulation program with the ability to determine the existence or non-existence of these intervisibilities. The method used in WHERSM is to maintain a set^(***) of "intervisibility snapshot matrices", I_{ijt} ($i = 1, 2, \dots, n$; $j = 1, 2, \dots, n$;

(*) Four-point interpolation could have been used here, but is very time consuming. And since it was determined that no significant further loss of accuracy occurs with the "nearest point" method, that one was implemented.

(**) See p.28 of "First Interim Progress Report on NBS Project WHERE in Support of USAAMCA Project MNPLS", NBS Report 10663, December 1971.

(***) Obtained from the Electromagnetic Compatibility Analysis Center, Annapolis, Md.

$t = 0, 20, 40, \dots, 120$) where $I_{ijt} = 1$ if unit j can receive and report a ranging signal from unit i at time t minutes into the simulation, and $I_{ijt} = 0$ if it cannot. Furthermore, in order to avoid drastic changes in the intervisibility pattern at the discrete times indicated by values of the subscript t , each row of a particular intervisibility matrix defined by a specific value of t , say t' , is overlaid with the corresponding row of the next-in-sequence matrix (defined by $t' + 20$) at some randomly chosen time between t' and $t' + 20$. (These times are chosen independently for each row and for each value of t .) More detail regarding each aspect of range computation will be found in sections 3 and 5.

The last item of concern for the MNPLS simulation is that of position estimation. Although that item is an integral part of any simulation study of this nature (indeed, the most important part), no discussion of the various algorithms used in WHERSM is provided here, since the project staff members responsible for the development and implementation of position location algorithms have presented such a discussion in Working Paper Number 7.

In order to provide the reader with a better understanding of the flow of events within WHERSM, and to facilitate further discussion of the WHERSM program, a flowchart is given in figure 1 which indicates the basic logic of the simulation. With one exception (see section 9), each box in the flowchart corresponds to an easily recognizable module within the WHERSM code listed in appendix B. In every case, entry to box 2 is made just after the simulation's current time is incremented by a constant value. That constant, Δt , is referred to as the subcycle time and is the time during which all the operations required to locate a given unit (one completion of the loop formed by boxes 2 through 7 of the flowchart) must be performed. Cycle time is the time required to locate each unit at least once. This may be larger than $n \cdot \Delta t$, where n is the total number of units, since certain faster moving units may need to be located more often

than do slow moving units. Locatee refers to that unit which, at any instant of simulated time, is in the process of being located, and locator refers to a unit whose range to the locatee is used to estimate the position of the locatee.

The remainder of this report discusses the workings of WHERSM at a more detailed level. Section 2 explains Boxes 2 and 3 of the flowchart in figure 1, while Section 3 discusses Box 4. Section 4 explains Boxes 5, 6, and 7; Section 5, Boxes 8 and 9; Section 6, Boxes 10 and 11; Section 7, Box 12, Section 8, Boxes 13 and 14; and Section 9, Box 15. Section 10 contains a discussion of the input needed to make a run with the present version of the code; and Section 11 contains a discussion, with examples, of the outputs that will result from such a run. Appendix A is a dictionary of variables used in the program; Appendix B, a listing of the program; and Appendix C, a very detailed flowchart of the program.

Although the remaining section headings refer directly to the flowchart in figure 1, the content of those sections can be better understood by referring to Appendices B and C.

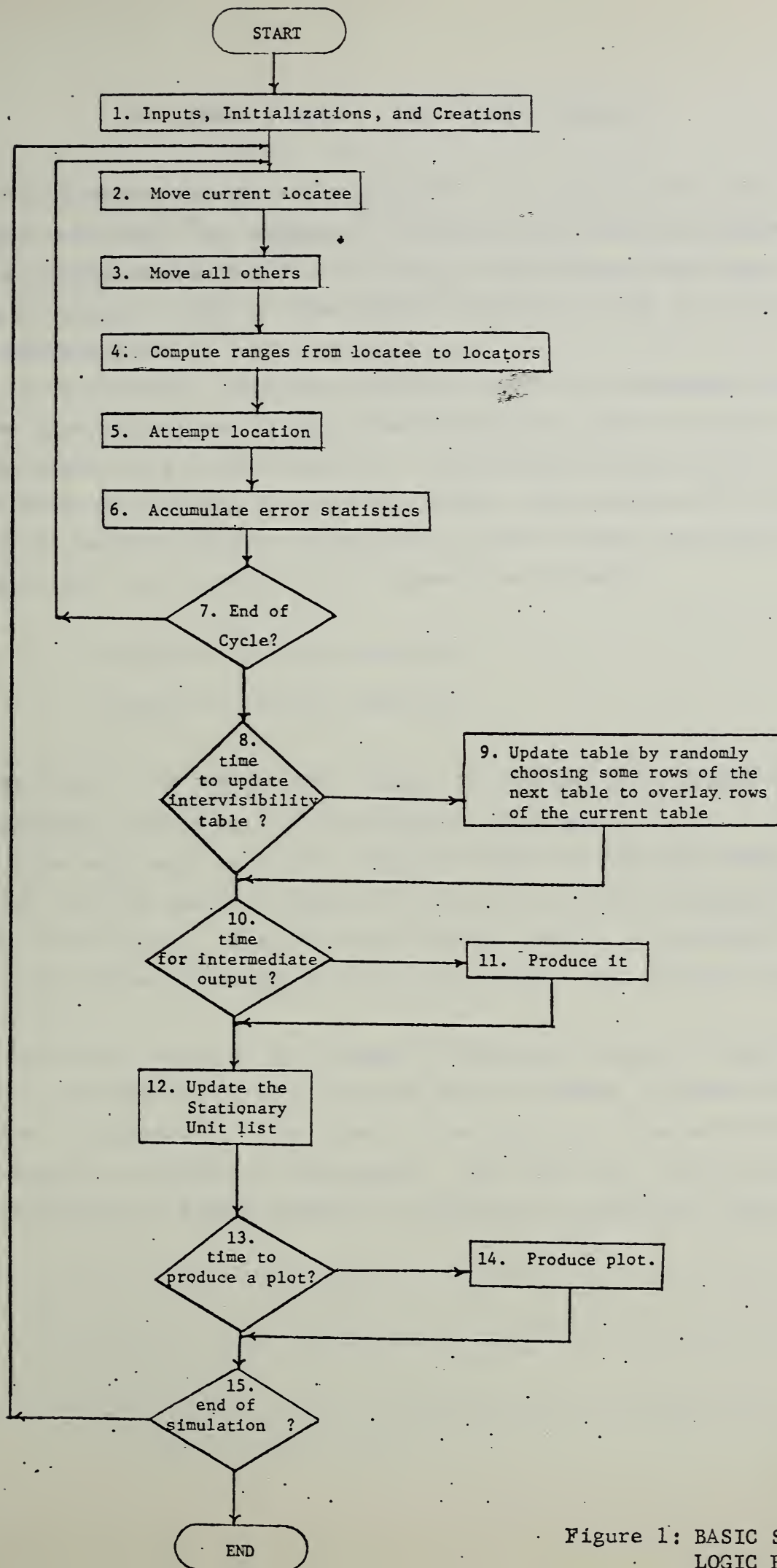


Figure 1: BASIC SIMULATION
LOGIC FLOWCHART

2. BOXES 2 and 3: THE MOVEMENT MODULES

The movement modules are responsible for computing the true positions of each unit every Δt seconds. In the current version of WHERSM, this is a two-stage process; i.e., x and y are computed, and then z is computed using x and y . The method by which x and y are computed is explained first.

In the xy -plane, units are assumed to move in a piecewise linear manner (the projections of the three-dimensional curves of motion onto the xy -plane are piecewise linear). The piecewise linear motion is effected by maintaining a matrix of current true positions (TP) and a matrix of current velocity vectors (VV), both of which are initialized by card input, and by updating TP linearly as follows:

$$TP(I,1) = TP(I,1) + VV(I,2)$$

$$TP(I,2) = TP(I,2) + VV(I,2)$$

for each unit $I = 1, 2, \dots, NU$, where NU is the total number of units. See Appendix A for a detailed explanation of TP and VV.

After each unit is moved, a check is made to determine whether it is time for that unit to change its direction of motion (change times are stored in $VV(I,3)$). If so, the new velocity vector is retrieved from the matrix $VV1$ where all velocity vectors are stored, and replaces the existing one in VV.

The second stage of the movement computation, that of computing z , has been implemented in two different ways in WHERSM, although only one appears in appendix B. (The other is too specific to include here and, furthermore, is trivial to implement.) The first one, the one not included here, is simply a matter of coding up a continuous function

representation^(*) for the z coordinate and computing a value of z once x and y have been determined.

The method of determining z that is included in Appendix B is somewhat complicated, in that it involves a preprocessor, overlays, a mass storage retrieval routine, and parallel processing. The preprocessor is a routine called BUILD, whose function is to read a tape containing topographical data in the format designed by the Electromagnetic Compatibility Analysis Center^(**), decode and unpack the individual altitudes above sea level that are given on that tape, write the individual elevations out on the drum, create an index (NDX) to the information on the drum, and initiate the overlaying of itself by the rest of the WHERSM code.

The subroutine that actually provides the z coordinate, HEIGHT, is mainly a drum retrieval routine. HEIGHT takes the x and y that are input to it, computes the nearest available grid point to that x and y, determines the position on the drum of the altitude associated with that grid point, and initiates the retrieval of that altitude. This last point is important -- HEIGHT only initiates the read. It does not wait for the completion of that operation, but instead immediately returns control to the calling routine, which has the option of waiting for the completion of the retrieval or of performing other tasks that are not dependent on knowledge of the new z coordinate. In view of the fact that the time required to retrieve one elevation from the drum is approximately 4.254 milliseconds, the above parallel processing capability provided by the HEIGHT routine allows considerable time savings to be

(*) Note that obtaining such a function for "fictitious" terrains is rather easy, but that surface fitting for "real" terrains may not be trivial. See, e.g., Pavilidis, T., "Piecewise Approximations of Functions of Two Variables and its Application in Topographic Data Reduction", Princeton University, Tech. Report No. 86, Sept. 1970, and "A Program Implementing Piecewise Linear Approximations of Functions of Two Variables", Princeton University Tech. Memo. No. 3, March 1971.

(**) Encoded altitudes above sea level at points that are three seconds of latitude and longitude apart.

achieved through the thoughtful organization of the main routine relative to height retrieval.

The previous comment regarding organization serves to introduce the last point to be discussed in this section -- the reason for including two movement modules -- one (Box 2) for the current locatee, and one (Box 3) for all other units in the field. Although all units move every subcycle, a new height is determined only when a unit serves as locatee. (In between, heights are treated as remaining constant.) Hence, while the height for the locatee is being retrieved from the drum, the program can continue on with the moving of all other units, thus taking advantage of the parallel processing provided by HEIGHT. There is very little loss of "realism" with this approach, since due to the design criteria no unit moves farther than 44 meters between times when it is a locatee, and further, since the topographical grid points define a rectangle with dimensions approximately 68m x 92m. Hence, only a small distortion of an already grossly simplified terrain structure occurs.

There is yet another reason for separating movement into two modules. Namely, it reduces the effort involved in implementing a very specific type of movement scenario. For example, the code in Appendix B includes a technique for keeping units from spreading out too much, by disallowing movement past a boundary that moves at a speed determined by input. (This is convenient when a randomly generated scenario that includes foot soldiers and ground vehicles is used. By letting the boundary move at foot soldier speed, the ground vehicles can be kept "close by".) In this case, it is wasteful to monitor a unit's movements all the time. To check only when a unit serves as locatee is sufficient. And this of course is easier and faster when there is a separate movement module for the locatee.

3. BOX 4: THE RANGE COMPUTATION

The range computation module is responsible for computing reported ranges by computing true ranges and perturbing them, and for determining whether each range should be reported. The computation of true ranges involves only Euclidean distance calculations, and the perturbation values are obtained by a call to subroutine RG. Subroutine RG generates pseudo-random numbers having a Gaussian distribution with mean zero and variance $36m^2$, and then randomly chooses 1% of them to which it imputes a positive bias by adding 7 meters to each. Except for the biasing procedure, the method used to generate these numbers is the same as the one proposed by A. Rotenburg^(*). RG calls RGU which provides uniformly distributed pseudo-random numbers, generated by the multiplicative congruential method. Other distributions, or other parameter values in these, could easily be substituted in these routines. In order to save time, since RGU is called so frequently during the simulation, Rit is written in SLEUTH, the machine language for the UNIVAC 1108. However, it is very short and can easily be rewritten in FORTRAN.

There are two items that determine whether a particular range should be reported. The first of these is the logical matrix RPT, which is initialized at input time, and which, for each unit, has the logical value "TRUE" if that unit is one that reports received ranges, and "FALSE" if it is not. Note that the maintenance of such a matrix allows the testing of scenarios in which the set of reporting units changes during a run, and where such changes are either exogenous or endogenous events.

The second item that determines whether a particular range should be reported is the matrix of intervisibilities (IT), which records whether or not a radio line of sight (LOS) exists between each pair of units

(*) Rotenburg, A., "A New Pseudo-Random Number Generator", J. ACM 7. (1960), pp. 75-77.

identified by the subscripts for IT. Also, each row of IT is condensed so that 36 row entries fit into one computer word, thereby greatly reducing storage use. Therefore, the only concern of the range computation module regarding intervisibilities is that of locating and retrieving a particular entry in IT from a block of 36 entries, which informs it whether the range between the defining units should be calculated.

In summary, then, for each subcycle the range computation module steps through a list of the units (excluding, of course, the one currently being located) and checks RPT to see first if a particular unit is a reporter. If so, IT is checked to determine whether that unit can "see" the locatee. If that also is true, then the true distance is calculated, RG is called to obtain a perturbation value, and that value is added to the true distance yielding a reported range. Reported ranges are stored in the matrix RANGE, with the unit numbers of the units that reported each range in matrix LOCATR.

4. BOXES 5, 6, and 7; THE LOCATION ATTEMPT, STATISTICS ACCUMULATION, AND "END OF CYCLE" TEST

The discussions of Boxes 5, 6, and 7 have been combined into this one section simply because there is very little to say about any of them.

The location attempt consists of a call to subroutine FINDIT, which is a preprocessor for the actual location algorithms. As was mentioned in Section 1, no discussion of the algorithms is provided here; nevertheless, some discussion of the input to the algorithms from WHERSM should be given. All input to the position location algorithms is achieved via the labeled common block CB1, which consists of the following variables: EP, LOCATR, RANGE, S, TLOC, W, IU, CLK, XNEW, YNEW, ZNEW, FXD, RPT, and NTYPE. For a description of each of these variables, see Appendix A.

The error statistics that are accumulated are: the current coordinate errors (coordinate-wise subtraction of true from estimated positions), the locatee's position location error (Euclidean distance between true and estimated positions in xy-plane), the error in the z coordinate, the max error in the xy-plane over all cycles and over the current block of KIC cycles (KIC is an input parameter), the maximum error in z over all cycles and over the current KIC cycles, and the time to perform the location operation (obtained by interrogating the computer's internal clock). The statistics accumulated here are used to produce the intermediate output explained in Section 11.

The "end of cycle" test (Box 7 of the flowchart in figure 1) is nothing more than the end of a DO loop which steps the simulation program through each one of the subcycles in a cycle.

5. BOXES 8 and 9: THE "INTERVISIBILITY TABLE UPDATE"

The "intervisibility update" module is concerned with maintaining an LOS pattern that is checked by the WHERSM program during range computation (see Section 3). The "current" intervisibility pattern is kept in the matrix IT, which is automatically updated in a manner determined by parameters that are input to the simulation program. This section explains the manner in which IT is updated.

The intervisibility snapshot matrices that have been discussed in Sections 1 and 3 are read in from tape at input time and stored in the matrix ITB. (The format of the tape is explained in Section 10.) At the start of a simulation run, IT is initialized to the first of the snapshot matrices--the matrix that corresponds to a line of sight snapshot taken at time zero. Every K2C cycles thereafter, K3C randomly chosen rows of IT are overlaid with the corresponding rows of the snapshot matrix that is next in sequence. (K2C and K3C are input parameters; see Section 10.) This overlaying process continues until simulated time has been advanced to the time at which that next snapshot was taken, at which point the intervisibility update module completes the overlaying of all rows not previously updated, and prepares to start over again with the new matrix and the one after that.

The choice of a row to be updated is achieved by choosing a pseudo-random number, call it I, between 1 and NU, where NU is the number of units in the field and also the number of rows and columns in IT. If the corresponding row of IT has not previously been updated (determined by a check of the logical matrix ITIND), it is then updated. If that row has been updated, try row I+1. If that row has been updated, try row I+2. This search process continues until either an "updatable" row is found or until the checking process circles back to row I. (This last comment implies, of course, that row NU+1 is considered to be row 1.) If no updatable row is found, nothing changes and the "intervisibility update" module has completed its task.

6. BOXES 10 and 11: THE PRODUCTION OF INTERMEDIATE OUTPUT

This section of code controls the production of intermediate error output as portrayed in figure 3, and which is discussed in Section 11.

Basically, this module's function is to convert the statistics accumulated in Box 6 to a form that is more meaningful for output. For example, sums of previous errors are divided by the appropriate number to obtain averages, coordinate error counts are made to produce quadrant counts, etc.

The frequency with which the intermediate error output is produced is controlled by an input parameter called K1C. The output summary is provided at the end of every K1C cycles.

7. BOX 12: UPDATE THE STATIONARY UNIT LIST

The stationary unit list (logical matrix FXD) is a list that contains "known" information regarding which units are moving and which units are not moving throughout the simulation. By "known" is meant that this information is available to the position location algorithms.

There are two types of real world scenarios that the matrix FXD has been instrumental in simulating. The first of these is the situation where units stop and start throughout the "battle", and send a signal back to the master computer indicating that they have just stopped or that they have just started moving. The second scenario is one in which certain units are "surveyed in" and never move, whereas no other units are ever treated as fixed. Knowledge of these facts is very helpful to the position location algorithms, but the "how" and the "why" of that helpfulness will not be discussed here. For that information, the reader is referred to Section 3 of the parent document to which this Working Paper is an appendix.

In the current version of WHERSM, the starting and stopping of units mentioned above is achieved by maintaining the matrix FXD, and by updating it in a random manner that is controlled by parameters which are input to the simulation. For each unit, FXD is "TRUE" if that unit is currently fixed, and "FALSE" if that unit is currently moving. The input parameter PSS(2) is the average time a unit remains fixed once it stops moving, while PSS(1) is the average time it keeps moving once it starts moving. These times are converted to probabilities, stored back into PSS(1) and PSS(2), and those probabilities in turn are used to determine, at each cycle, whether a unit should have its state ("fixed" or "moving") changed.

The determination of state changes is precisely the purpose of the "stationary list update" module. It is achieved by obtaining a uniformly distributed (0,1) pseudo-random number from the internal subroutine RU, and, if that number is less than the appropriate PSS probability value, changing the state of that particular unit.

It should be noted that in the second scenario mentioned above, one of course does not want the stationary unit list to be changed at all during the simulation. This can be guaranteed simply by letting the values of PSS(1) and PSS(2) at input time be very large numbers. It should also be noted that FXD is initialized at input time.

8. BOXES 13 and 14: PRODUCE THE X-Y PLOTS

This module is very small, consisting mainly of two calls to subroutine GRAPH. The first call produces x-y plots of the true positions of all the units, and the second call produces x-y plots of the estimated positions of all the units.

There are eight parameters in the call statement for subroutine GRAPH: N, X, Y, IWRD, XMAX, XMIN, YMAX, and YMIN. N is the number of points to be plotted, X and Y are vectors of the x and y coordinates of the points to be plotted, IWRD contains either 'TRUE' or 'EST.' depending on whether the points to be plotted are true or estimated positions, and XMAX, XMIN, YMAX, and YMIN are the upper and lower limits of the axes. Those last (XMAX through YMIN) are part of the input to the simulation program.

For a description of the plots that are produced, see Section 11.

The frequency with which the plots are produced is controlled by the input parameter K4C; the plots are produced after every K4C cycles.

9. BOX 15; THE END OF SIMULATION

Box 15 of the flowchart in figure 1 is the only one that does not correspond to an easily recognizable module in the WHERSM code. Rather, its function is performed by a set of IF statements scattered throughout the other modules. The IF statements are all identical; they test whether the number of cycles required to run the simulation for the length of time indicated by the input parameter SIMTIM have been completed. If so, then a final set of "intermediate" output is produced before termination.

10. THE INPUT TO THE SIMULATION

There are three types of input used in the WHERSM program: input from cards, tape input, and "hardwired" input in the form of DATA and PARAMETER statements inserted in the program deck. This section will discuss each type of input with regard to input values and formats. Card input is explained first.

The "Catch-All" Cards: These cards contain the various parameters and constants needed by the program; there are two of these cards, and the table below explains their contents. For a definition of the variables mentioned, see Appendix A.

	<u>Columns</u>	<u>Format</u>	<u>Variable Name</u>
Card No. 1:	1-5	I5	NU
	6-10	I5	NS
	11-15	I5	K1C
	16-20	I5	K2C
	21-25	I5	K3C
	26-30	I5	K4C
	31-35	I5	INR
	36-40	I5	NSP
	41-50	F10.5	PSS(1)
	51-60	F10.5	PSS(2)
	61-70	F10.5	BND
	71-80	F10.5	BNDSP
Card No. 2:	1-10	F10.5	CYCTM
	11-20	F10.5	SIMTIM
	21-30	F10.5	XMAX
	31-40	F10.5	XMIN
	41-50	F10.5	YMAX
	51-60	F10.5	YMIN

The "Reporting Units" Cards: These cards contain the unit numbers of the units that will be used at the start of the simulation as reporters. More specifically, the elements of the matrix RPT that correspond to the numbers that appear on the Reporting Units Cards are the only elements that are initialized to the logical value "TRUE". All others are initialized to "FALSE". There are 20 numbers per card, each with a format of I4. The number of unit-numbers that appear on these cards is the value of INR (see Catch-all cards above). Consequently, there will be $[(\text{INR}-1)/20] + 1$ Reporting Units cards.

The "Fixed Units" Cards: These cards contain the unit numbers of the units that will be used at the start of the simulation as fixed units. The elements of the matrix FXD are initialized in the same manner as are the elements of the matrix RPT above. The format for each card again is 20I4, and there will be $[(\text{NSP}-1)/20] + 1$ of these cards.

The "Type" Cards: The type cards contain the type number for each unit in the simulation. There currently are four unit types:

<u>Type Number</u>	<u>Type</u>	<u>Speed</u>
1	Foot Soldier	1 1/2 - 2 km/h
2	Ground Vehicle	8-10 km/h
3	Low Performance Aircraft	161 km/h
4	High Performance Aircraft	322 km/h

There will be exactly NU of these type numbers each with a format of I4, and therefore $[(\text{NU} - 1)/20] + 1$ Type cards.

The "Movement" Cards: These cards contain all the information needed by the program so that it can provide for the movement of units. There will be one of these cards for each unit in the scenario, hence NU cards in all. Their contents are as follows:

<u>Card Columns</u>	<u>Format</u>	<u>Meaning</u>
1-3	I3	Unit number
4-21	3A6	Unit ID (anything)
		Starting latitude:
22-24	I3	Degrees
25-26	I2	Minutes
27-28	I2	Seconds
		Starting longitude:
29-31	I3	Degrees
32-33	I2	Minutes
34-35	I2	Seconds
37-40	F4.0	Time ^(*) (hrs.)
41-44	F4.0	Azimuth (deg.)
45-48	F4.0	Speed (km/h)
50-53	F4.0	Time ^(*) (hrs.)
54-57	F4.0	Azimuth (deg.)
58-61	F4.0	Speed (km/h)
63-66	F4.0	Time ^(*) (hrs.)
67-70	F4.0	Azimuth (deg.)
71-74	F4.0	Speed (km/h)

There are two kinds of tape input used in the WHERSM program. The first of these is the tape that contains the height information in the form of a digitized terrain map. The format of this tape is as appears in figure 2, but no further discussion of format will be given here since the terrain tape was obtained directly from the Electromagnetic Compatability

(*) By "time" is meant the length of time during which the immediately following azimuth and speed are used. Control is passed to the next set when "time runs out" for this set.

Analysis Center which has published a report^(*) that documents such tapes in detail. The terrain tape must be mounted on logical unit number 8.

The other tape used as input to WHERSM is one that is mounted on logical unit number 7 and contains the intervisibility snapshot matrices discussed in Sections 1 and 3. It is a binary tape, created as follows: Let ITB be a three-dimensional matrix that contains the set of "packed" intervisibility snapshot matrices;

$$ITB(I,J,1), \quad I = 1,2,\dots,NU; \quad J = 1,2,\dots, [(NU - 1)/36] + 1$$

is the snapshot matrix taken at the first discrete time value; ITB(I,J,2) is the matrix taken at the second discrete time value; etc. Then the tape should have been created using a write statement of the form:

```
WRITE (7) (((ITB(I,J,K), J = 1, NITCOL), I = 1,NU), K = 1,K1)
```

where $NITCOL = [(NU - 1)/36] + 1$ (implying of course that 36 entries of each row of the 2 dimensional matrices are packed into one computer word), and K1 equals the number of snapshot matrices that exist.

The third and last type of input required to make a simulation run is the type that must physically be inserted in the program deck - the PARAMETER and DATA statement cards.

The table below lists these. The reader is referred to Appendix A for their meanings.

(*) Clemmitt, M. and Stone, R., "The Topographic Information System", Tech. Note No. ECAC-TN-007-222, June 1969, Electromagnetic Compatibility Analysis Center, Annapolis, Md.

<u>Variable Name</u>	<u>Type of Statement Contained In</u>	<u>Routine Contained In</u>
MNLAT	DATA	BUILD
MXLON	DATA	BUILD
TPOUT	DATA	WHERSM
IFREQ	DATA	WHERSM
CNX	DATA	HEIGHT
CNY	DATA	HEIGHT
W	DATA	WHERSM
FXDINI	DATA	WHERSM
L1	PARAMETER	WHERSM
L2	PARAMETER	WHERSM
L6	PARAMETER	BUILD
		WHERSM
		HEIGHT
L7	PARAMETER	BUILD
		WHERSM
		HEIGHT

11. THE OUTPUT FROM THE SIMULATION

Figure 3 contains a sample of the intermediate output produced by WHERSM. This section explains the contents of that output sample.

The first line of output contains the cycle number of that cycle after which the summary was produced, the simulation clock time, the average real clock time the program took to provide for the processing of one subcycle, and the average time taken by the algorithms to perform one location operation.

Each row of the table contains statistics for the unit whose unit number appears in column 1. Columns 2-7 contain statistics accumulated over all cycles previous to the current one. They are, in order:

- the average error in the xy-plane,
- the average error in the z coordinate,
- the maximum error in the xy-plane,
- the cycle during which that maximum error occurred,
- the maximum z error that occurred, and
- the cycle during which that maximum z error occurred.

Columns 8-11 contain statistics accumulated over the previous K1C cycles. Those statistics are: the average x-y error, the average z error, the maximum x-y error that occurred, and the cycle during which that error occurred.

Columns 12-14 contain the individual coordinate errors (estimated minus true) that occurred the last time each unit was located prior to the printing of the intermediate output. Columns 15, 16, and 17 list the current values of the S, RPT, and FXD vectors respectively; and column 18 gives the current value of the W vector, the weight.

The last row of the table, which begins with the word "ALL", gives for a column of averages the average of the numbers appearing above it; and for a column of maximums, the maximum of the maximums in the column.

CYCLE NUMBER 140. TIME = 4199.759 S. AVERAGE STIMULATION SUPRCYCLE TIME = .0114 S. AVERAGE ALGORITHM SUPRCYCLE TIME = .0063 S.

UNIT	XY AVG	ERROR STATS. Z AVG	OVER ALL XY MAX	140 CYCLES Z MAX	CYC	ERROR STATS. OVER LAST 10 CYCLES XY AVG	XY MAX	CYC	CURRENT EX-TX	COORDINATE EY-TY	ERRORS EZ-TZ	SRF	WEIGHT		
1	.24414+01	.91294+00	.91909+01	25	.16242+01	34	.46277+01	.50339+00	.78275+01	135	.284+01	-.124+01	-.435+00	TFE	.14161-01
2	.35454+01	.85762-01	.13054+02	25	.15272+00	38	.48220+01	.38292-01	.67530+01	137	.264+01	-.363+01	-.561-01	TFE	.11573-01
3	.20537+01	.91748+00	.10124+02	105	.13243+01	111	.25380+01	.13243+01	.25380+01	131	.104+01	-.164+01	-.132+01	TTT	.26447-01
4	.24157+01	.11534+02	.24763+01	137	.13297+02	116	.42995+01	.10892+02	.94783+01	137	-.994+00	-.295+00	-.104+02	TFE	.16089-01
5	.04719+01	.10120+01	.12016+02	99	.11822+01	28	.39797+01	.11395+01	.73595+01	137	-.926+00	-.203+01	-.116+01	TFE	.24585-01
6	.30024+00	.27290+01	.88017+01	122	.28855+01	1	.39415+01	.12489+01	.71701+01	132	.170+01	.231+00	-.229+01	TFI	.25712-01
7	.22493+01	.19100+00	.10547+02	118	.41177+00	09	.12492+01	.24232+00	.12492+01	131	.138+00	-.124+01	-.142+00	TTT	.25724-01
8	.21102+01	.99290-01	.86535+01	58	.21050+00	04	.16679+01	.21050+00	.16679+01	131	.111+01	-.125+01	-.231+00	TTT	.27179-01
9	.22773+01	.39777+01	.95356+01	108	.41296+01	85	.30588+01	.32576+01	.47633+01	133	.288+01	-.123+01	-.325+01	TFI	.23854-01
10	.21455+01	.16734+01	.11492+02	48	.19508+01	67	.12995+01	.19608+01	.12905+01	131	.129+01	.101+00	-.196+01	TTT	.27841-01
11	.25133+01	.15703+01	.94925+01	123	.30024+01	136	.34411+01	.29555+01	.74786+01	131	.384+00	-.217+01	-.300+01	TFI	.20555-01
12	.25234+01	.37031+01	.10452+02	6	.54271+01	140	.38779+01	.52943+01	.41314+01	131	.353+01	-.694+00	-.543+01	TFE	.10270-01
13	.27175+01	.25427+01	.09748+02	09	.29795+01	118	.24089+01	.29705+01	.24089+01	131	.152+01	-.187+01	-.298+01	TTT	.26687-01
14	.14774+01	.79410+01	.13946+02	30	.85953+01	08	.37038+01	.67913+01	.79643+01	139	.420+01	-.394+01	-.702+01	TFE	.13082-01
15	.22230+01	.85750+00	.10457+02	121	.16539+01	1	.40113+01	.13546+01	.82947+01	136	.298+01	-.170+01	-.155+01	TFE	.10453-01
16	.25355+01	.64727+00	.14332+02	104	.15000+01	1	.46227+01	.76429-01	.71994+01	136	.409+01	-.107+01	-.691-01	TFE	.13824-01
17	.24461+01	.55455+00	.10144+02	129	.26535+01	140	.43296+01	.23610+01	.87303+01	136	.299+01	-.841+00	-.265+01	TFE	.17387-01
18	.15675+01	.12135+01	.99921+01	136	.86027+01	140	.19873+01	.26083+01	.88921+01	136	-.103+01	.164+01	-.264+01	TFE	.14204-01
19	.24492+01	.14792+01	.11560+02	33	.68025+01	1	.54770+01	.52097+01	.10985+02	133	.273+01	-.234+01	-.744+01	TFE	.16034-01
20	.14833+01	.43086+00	.13031+02	101	.435675+01	135	.30819+01	.39595+01	.90679+01	132	-.153+00	-.161+01	-.437+01	TTI	.26441-01
21	.27367+01	.42527+01	.12970+02	76	.79752+01	71	.42305+01	.65653+01	.42305+01	131	.421+01	.465+00	-.657+01	TTT	.27685-01
22	.20509+01	.42597+01	.15625+02	53	.15755+02	13	.12305+01	.33493+01	.12305+01	131	.114+01	-.457+00	-.335+01	TTT	.27451-01
23	.24159+01	.32265+01	.12333+02	74	.10189+02	77	.35549+01	.34079+01	.78397+01	137	.264+00	-.195+01	-.335+01	TFE	.83285-02
24	.24342+01	.27631+01	.14701+02	131	.13077+02	73	.51618+01	.58671+00	.14701+02	131	.464+01	-.892-01	-.370+00	TFE	.12657-01
25	.24122+01	.17163+01	.11560+02	113	.74756+01	62	.43317+01	.55699+00	.43318+01	131	.429+01	-.631+00	-.557+00	TTT	.27585-01
26	.20103+01	.15945+01	.14317+02	139	.21374+01	20	.63410+01	.10169-01	.14317+02	138	.382+01	-.626+01	-.465-01	TFE	.21686-01
27	.20021+01	.35683+01	.14632+02	105	.91285+01	14	.39969+01	.64352-01	.79851+01	137	-.125+01	-.335+01	-.504-01	TFE	.19767-01
28	.23005+01	.23007+01	.13544+02	33	.10227+02	25	.29465+01	.16676+01	.29465+01	131	.285+01	-.392+01	-.147+02	TFE	.27650-01
29	.19225+01	.55312+01	.14575+02	73	.14732+02	140	.38722+01	.13192+02	.82939+01	140	.178+00	-.392+01	-.147+02	TFE	.20070-01
30	.24025+01	.11546+01	.12351+02	130	.44768+01	135	.50031+01	.36652+01	.12265+02	135	.375+01	-.687+01	-.410+01	TFE	.11522-01
31	.24025+01	.44557+01	.12022+02	101	.11915+02	103	.35341+01	.11557+02	.10252+02	138	-.105+01	-.396+00	-.105+02	TFE	.16541-01
32	.20025+01	.70161+01	.11918+02	35	.11451+02	29	.39844+01	.01909+01	.38284+01	131	.354+01	-.146+01	-.820+01	TTT	.27611-01
33	.24025+01	.30115+01	.11918+02	70	.19447+02	77	.16223+01	.11483+02	.10166+02	140	.093+01	.467+01	-.109+02	TFE	.19610-01
34	.21122+01	.52529+01	.15604+02	112	.22580+02	118	.15080+01	.19463+02	.18785+01	131	.205+00	-.159+01	-.105+02	TTT	.27351-01
35	.27454+01	.70747+00	.15030+02	139	.33704+01	54	.53590+01	.55589-01	.15939+02	138	.245+00	-.301+01	-.220+00	TFE	.14553-01
36	.24039+01	.19790+01	.15338+02	15	.95032+01	140	.38359+01	.81762+01	.94795+01	135	.271+01	-.275+01	-.950+01	TFE	.15609-01
37	.24031+01	.55750+01	.16954+02	24	.27192+02	140	.53173+01	.20046+02	.12370+02	139	-.105+01	-.222+01	-.255+02	TFE	.11397-01
38	.24031+01	.47746+01	.24434+02	71	.20080+02	67	.61602+01	.86331-01	.16866+02	139	-.618+00	-.200+01	-.450+00	TFE	.11193-01
39	.24031+01	.35272+01	.35762+01	85	.66419+01	86	.54237+00	.66419+01	.54237+00	131	-.387+00	-.380+00	-.664+01	TTT	.27683-01
40	.24145+01	.31754+00	.14432+02	134	.21461+01	136	.47178+01	.14505+01	.14432+02	134	.274+01	-.218+01	-.882+00	TFE	.22936-01
ALL	.29795+01	.29773+01	.24436+02	71	.27189+02	140	.36377+01	.43993+01	.16866+02	139					

QUADRANT COUNTS OF ERRORS

1 0 4 8 27

The next item of output is the quadrant counts of errors. This is a display of the directions of errors. It is obtained from columns 12 and 13 of the table by counting the number of entries in those columns (by their algebraic signs) that fall in each quadrant.

The last items of output are the xy-plots, an example of which is given in figure 4. The two lines at the top of the graph state whether the graph is of true or estimated positions, list the unit numbers of the units that appear on that graph, and list the plotting symbols used to represent those units. The letters of the alphabet are used as plotting symbols, so that if more than 26 units are in the simulation, a separate graph will be produced for every 26 (or fraction thereof) units. If more than one unit occupies the same graph position, a number will appear in that graph position indicating the number of units in that position. The axis ranges are controlled by XMAX, XMIN, YMAX, and YMIN, which are part of the input to the simulation, and if points fall beyond those boundaries an annotation to the graph will appear indicating the occurrence of that situation.

THE ASSOCIATED SYMBOLS ARE A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

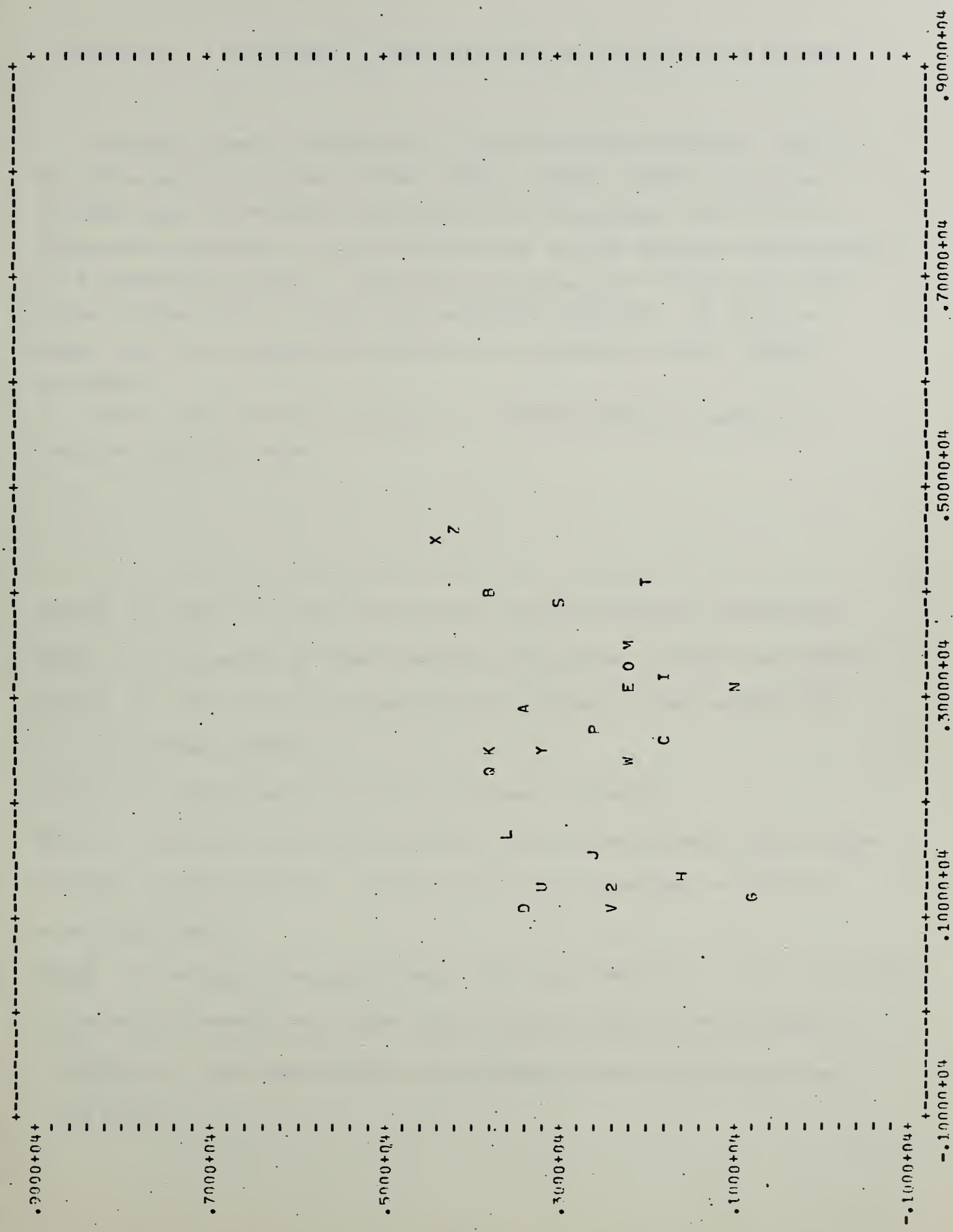


Figure 4: Example of XY-Plots

APPENDIX A: A DICTIONARY OF VARIABLES USED IN THE SIMULATION PROGRAMS

In this appendix, definitions of all non-trivial variables used in the three main simulation routines (BUILD, WHERSM, HEIGHT) are given. In each case, the variable being defined is underlined, and if it is a dimensioned variable, it is written exactly as that variable should appear in a DIMENSION statement. Immediately following the variable name will appear either a B,W, or H (or any combination thereof). By this is meant that the variable being defined is a variable of BUILD, WHERSM or HEIGHT.

Note: All variables conform to the FORTRAN implicit naming conventions regarding mode.

* * * * *

AVASBT W. The sum of real times used to perform position estimations.

AVKTR W. The number of time intervals accumulated in AVASBT and AVSUBT.

AVSUBT W. The sum of real times used to simulate a "real world" sub-cycle (includes AVASBT).

AZ(4) W. Buffer used in reading in movement azimuths.

BND W. Boundary beyond which units are not allowed to move. Unit of measurement is the kilometer. If this option is not desired, set BND to some large number.

BNDSP W. On input, the speed (km/h) with which BND moves in both directions. Immediately converted to a value that is added to BND at every subcycle.

C(14641) B. An output buffer that contains decoded elevations above sea level.

CLK W. The simulation's clock. Contains simulated time in seconds.

CNX H. The inverse of the average longitudinal distance (in meters) between grid points on the terrain tape.

CNY H. Same as CNX but for latitude.

CT(4) W. Input buffer used to read in the length of time a unit is to move according to a particular azimuth and speed.

CYCTM W. The "real time" length of a cycle in seconds.

D(14641) B. Output buffer that contains decoded altitudes above sea level.

DELTA W. "Real time" length of a subcycle.

DX(L1) W. List used to store the difference in the x-coordinates of the true and estimated positions. Computed immediately after a unit has had its position estimated.

DY(L1) W. Same as DX, but for y-coordinates.

DZ(L1) W. Same as DX, but for z-coordinates.

EP(L1,14) W. Contains estimated positions for each unit, in addition to some data relevant to those positions. EP(I,1-3) is the estimated position of unit I which was determined the next-to-last time it served as locatee; EP(I,4-6), the last time it served as locatee. The rest of the EP matrix is used to manipulate positions when the relocation option is used (see Working Paper No. 8).

ERR W. The error in position estimation in the xy-plane and then in the z-coordinate. The Euclidean distance between true (TP) and estimated (EP) positions is used as the error.

IFY H. Same as IFX, but for the y-coordinate. Units of measurement here are 3 seconds of latitude.

IGP W. Pointer to the unit with the largest xy-error over all cycles.

IGT W. Pointer to the unit with the largest xy-error over the last K1C cycles.

INR W. Initial number of units allowed to report.

IP W. Pointer used in constructing VV1.

IPX H. The x-coordinate of some other position (for which a height is already available), after it has been converted to the units of the terrain.

IPY H. Same as IPX, but for y-coordinate.

IQ0 W. Number of units whose estimated position is exact.

IQ1 W. The number of units whose estimated position is in error in the direction of the first quadrant.

IQ2 W. Same as IQ1, but for quadrant 2.

IQ3 W. Same as IQ1, but for quadrant 3.

IQ4 W. same as IQ1, but for quadrant 4.

IQUAN B. Quantization factor for the block of terrain grid points currently being decoded. (See ECAC-TN-007-222, op.cit.)

IR W. Counter for the number of ranges computed in a particular subcycle.

IT(L1,L3) W. The intervisibility table currently being used.

ITB(L1,L3,7) W. The set of all intervisibility tables.

ITIND(L1) W. Logical matrix which indicates for each row of IT, whether that row has been updated yet.

IU W. The unit number of the unit currently serving as locatee.

IV W. The (0-1) answer to the question of existence of an LOS between the pair of units in question.

IVMAT W. Pointer to the intervisibility matrix currently being used to update IT.

IZT W. Pointer to the unit with the largest z-error over all cycles.

K1C W. The number of cycles after which intermediate output is to be produced.

K1CC W. Counter used with K1C.

K2C W. The number of cycles after which an intervisibility update is to occur.

K2CC W. Counter used with K2C.

K3C W. The number of rows of IT to be updated when an intervisibility update occurs.

K4C W. The number of cycles after which xy plots are to be produced.

K4CC W. Counter used with K4C.

KUF(L1) W. For $I = 1, 2, \dots, NS$, KUF(I) is the unit number of the unit which is to serve as locatee during the Ith subcycle.

KZ W. The seed value for the internal uniform psuedo-random number generator, RU.

L B. Status parameter for I/O processor NTRAN. L equals -1 until the completion of the I/O command.

L1 W. Parameter variable used in dimensioning. L1 must be greater than or equal to NU.

L2 W. Parameter variable used in dimensioning. $L2 \geq NS$.

L3-L5 W. Parameter variables used in dimensioning, but since they are strictly dependent on L1, they are automatically set.

L6 B,W,H. Conceptually organize all the 6' x 6' terrain blocks on the terrain tape according to the latitude-longitude of their southwest corner. L6, then, is the number of these blocks when counting from west to east.

L7 B,W,H. Same as L6, but count from south to north.

LOCATR(L1) W. The unit numbers of the units whose ranges are being reported in a given subcycle.

M W. Status parameter for I/O processor. See L.

M1 B. Status parameter for I/O processor. See L.

M2 B. Status parameter for I/O processor. See L.

MAXCYC W. The length of the simulation in terms of cycles.

MNLAT B. The minimum latitude (in seconds) for which a height is available on the terrain tape.

MXLON B. The maximum longitude (in seconds) for which a height is available on the terrain tape.

NBITS B. The number of bits into which the elevation in question is "packed".

NCYC W. The cycle number of the cycle currently being processed.

NDX(L6,L7) B,H. An index to terrain blocks that are written on the drum by BUILD. NDX(1,1) is the position of the 6' x 6' terrain block whose southwest corner is identified by (MNLAT, MXLON); NDX(1,2) is the position of block (MNLAT, MXLON + 3600); NDX(2,1) is block (MNLAT + 3600, MXLON); etc.

NITCOL W. The number of computer words in a packed row of the inter-visibility matrices.

NS W. The number of subcycles in a cycle.

NSP W. The number of units which, at the start of the simulation, will be identified as being stationary.

NTYPE(4) W. The unit type of each unit. If NTYPE(I) is 1,2,3, or 4, then unit I is respectively a foot soldier, a ground vehicle, a low performance aircraft, or a high performance aircraft.

NU W. The number of units in the simulation.

NUMEL B. The number of elevations per word for a given terrain block represented by a logical record on the terrain tape.

PSS(2) W. At input time, PSS(1) is the average time a unit keeps moving once it starts moving, and PSS(2) is the average time it remains fixed once it stops. Subsequently, they are the probabilities of changing state. If no random starting and stopping is desired, set both PSS(1) and PSS(2) to very large numbers on the catch-all-cards.

RADS W. Constant representing the number of degrees in a radian.

RANGE(L1) W. The ranges being reported for a given subcycle.

RCON W. Constant used in converting a speed and azimuth combination into direction cosines.

RPT(L1) W. Logical matrix that indicates for each unit, whether or not that unit is a unit that is allowed to report ranges.

S(L1) W. Logical matrix that indicates for each unit whether or not that unit was "successfully" located the last time a position estimation was attempted. S is set by the position location algorithms.

SIMTIM W. The "real time" length of the simulation, in minutes.

SP(4) W. Input buffer used in reading in speeds for each unit.

STAT1(L1,10) W. For each row: column 1 is the sum of position estimation errors (in xy-plane) over all cycles for the unit associated with that row; column 2, the maximum xy-error that occurred over all cycles for that unit; column 3, the cycle number of the cycle during which that maximum error occurred; 4, the partial sum of xy-errors over the current K1C cycles; 5, the maximum xy-error over the current K1C cycles; 6, the cycle number associated with that error; 7, the sum of the z-errors over all cycles for that unit; 8, the maximum z-error over all cycles; 9, the cycle number associated with that z-error; and 10, the partial sum of z-errors over the current K1C cycles.

STAT2(L1,2) W. Column 1 contains, for each unit, the number of times that unit was "successfully" located since the start of the simulation; and column 2, the number since the start of the current K1C cycles.

TLOC(L1,4) W. The clock time at which the estimated positions in EP were obtained.

TP(L1,3) W. The current true position of each unit. TP(I,1) is the x-coordinate; TP(I,2), the y-coordinate; and TP(I,3), the z-coordinate.

TPOUT W. Logical variable that allows for the writing of all intermediate output on logical unit 9, in addition to the standard output unit.

T1-T4 W. Parameters in calls to the system's clock. Used in determining AVASBT and AVSUBT.

VV(L1,4) W. Currently used velocity vector for each unit. Column 1 is the x component; column 2, the y component; 3, the time at which that vector should no longer be used; and 4, a pointer into VV1 where the next velocity vector to be used resides.

VV1(L5,4) W. The set of all velocity vectors for all units. The format of each vector is the same here as in VV, but the h ($h \leq 3$ currently) vectors for unit number 1 are in rows 1 through h of VV1, followed by the vectors for unit number 2, etc.

W(L1) W. Vector used by the position location algorithms to weight the various estimates (see Working Papers No. 7 and 8). W should be initialized to the weight to be assigned to non-fixed units at the start of the simulation.

XMAX, XMIN, YMAX, YMIN W. The upper and lower bounds of the x-and y-axes (in meters) to be used in producing the xy-plots.

ZPSUM W. The sum of the position estimation errors (in the z-coordinate)
over all units over the last K1C cycles.

ZTSUM W. The sum of the position estimation errors (in z-coordinate)
over all units over all cycles.

APPENDIX B: LISTING OF THE SIMULATION PROGRAM

- 1. CHN 1
- 2. SEG BUILD
- 3. CHN 2
- 4. SEG WHERSM

MAP LCC 1104 0036

MAIN PROGRAM

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000475
0000 *DATA 114647
0002 *BLANK 000073

EXTERNAL REFERENCES (BLOCK, NAME)

0003 NTRAN
0004 CHAIN
0005 NWDUS
0006 NI02\$
0007 NSTOP\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0000	114604	1F	0001	000021	10L	0001	000334	100L	0001	000035	127G	0001	000123	151G
0000	114613	2F	0001	000347	200L	0001	000362	210L	0001	000266	213G	0001	000372	220L
0001	000410	230L	0001	000420	240L	0000	114622	3F	0001	000031	30L	0001	000112	50L
0001	000161	68L	0001	000171	70L	0001	000255	80L	0001	000442	900L	0001	000452	910L
0001	000462	920L	0001	000471	930L	0001	000324	98L	0000	R	023420	C	0002	R
0002	R	000003	CNY	0000	R	060101	D	0000	I	114566	I	0000	I	011610
0000	I	114574	IBIAS	0000	I	114577	IBIT	0000	I	114601	IEL	0000	I	114562
0000	I	114571	J	0000	I	114564	K	0000	I	114565	L	0000	I	114570
0000	I	114600	M	0002	I	000000	MNLAT	0002	I	000001	MXLON	0000	I	114563
0000	I	114576	N	0000	I	114575	NBITS	0002	I	000004	NDX	0000	I	114603

00101	1*	PARAMETER L6=5,L7=11
00103	2*	PARAMETER L8=L6*L7
00104	3*	COMMON MNLAT,MXLON,CNX,CNY,NDX(L6,L7)
00105	4*	DIMENSION IA(5000),IB(5000),C(14641),D(14641)
00106	5*	DATA NDX/L8*-1/
00110	6*	DATA MNLAT,MXLON/2544,4302/
00113	7*	CALL NTRAN(8,22)
00114	8*	CALL NTRAN(35,22)
00115	9*	M2=0
00116	10*	K=0
00116	11*	C READ FIRST RECORD.
00117	12*	CALL NTRAN(8,2,5000,IA,L)
00117	13*	C WAIT FOR END OF TAPE READ.
00120	14*	10 IF (L+1) 20,10,30
00120	15*	C IF ERROR, 900. IF EOF, 200.
00123	16*	20 IF (L+2) 900,200,30
00126	17*	30 DO 40 I=1,L


```

00131 19* C INITIATE THE NEXT TAPE READ.
00132 20* C CALL NTRAN(8,2,5000,IA,L)
00133 21* C DECODE LATITUDE AND LONGITUDE OF FIRST TERRAIN BLOCK.
00134 22* C LAT=FLD(0,18,IB(3))
00135 23* C LON=FLD(18,18,IB(3))
00136 24* C COMPUTE ITS RELATIVE POSITION IN THE TERRAIN AND SET INDEX
00137 25* C TO THE DRUM.
00138 26* C I=(LAT-MNLAT)/6+1
00139 27* C J=(MXLON-LON)/6+1
00140 28* C NDX(I,J)=K*14641
00141 29* C DECODE NUMBER OF ELEVATIONS PER WORD, QUANTIZATION FACTOR,
00142 30* C BIAS, AND NUMBER OF BITS PER ELEVATION.
00143 31* C NUMEL=FLD(0,6,IB(5))
00144 32* C IQUAN=FLD(18,18,IB(5))
00145 33* C IBIAS=IB(6)
00146 34* C NBIITS=36/NUMEL
00147 35* C N=14
00148 36* C I=1
00149 37* C DECODE ELEVATIONS IN THE FIRST TERRAIN BLOCK.
00150 38* C 50 IBIT=0
00151 39* C DO 60 M=1,NUMEL
00152 40* C IEL=FLD(18,18,IB(N))
00153 41* C C(I)=(IEL*IQUAN+IBIAS)*.3048
00154 42* C I=I+1
00155 43* C IF (I.EQ.14642) GO TO 68
00156 44* C 60 IBIT=IBIT+NBIITS
00157 45* C N=N+1
00158 46* C GO TO 50
00159 47* C WAIT FOR CHANCE TO WRITE OUT THIS TERRAIN BLOCK.
00160 48* C 68 IF (M2+1) 69,68,70
00161 49* C 69 IF (M2+2) 910,920,70
00162 50* C WRITE IT OUT.
00163 51* C 70 CALL NTRAN(35,1,14641,C,M1)
00164 52* C BEGIN PROCESSING SECOND TERRAIN BLOCK. (PROCEDURE IS THE
00165 53* C SAME AS FOR THE FIRST BLOCK.)
00166 54* C K=K+1
00167 55* C N=IB(2)+4
00168 56* C LAT=FLD(0,18,IB(N))
00169 57* C LON=FLD(18,18,IB(N))
00170 58* C I=(LAT-MNLAT)/6+1
00171 59* C J=(MXLON-LON)/6+1
00172 60* C NDX(I,J)=K*14641
00173 61* C N=N+2
00174 62* C NUMEL=FLD(0,6,IB(N))
00175 63* C IQUAN=FLD(18,18,IB(N))
00176 64* C IBIAS=IB(N+1)
00177 65* C NBIITS=36/NUMEL
00178 66* C N=N+9
00179 67* C I=1
00180 68* C 80 IBIT=0
00181 69* C DO 90 M=1,NUMEL
00182 70* C IEL=FLD(18,18,IB(N))
00183 71* C D(I)=(IEL*IQUAN+IBIAS)*.3048
00184 72* C I=I+1
00185 73* C IF (I.EQ.14642) GO TO 98
00186 74* C 90 IBIT=IBIT+NBIITS
00187 75* C N=N+1

```



```

00225 76*      GO TO 80
00226 77*      9A IF (M1+1) 99,98,100
00231 78*      99 IF (M1+2) 910,920,100
00234 79*      100 CALL NTRAN(35,1,14641,D,M2)
00235 80*      K=K+1
00236 81*      GO TO 10
00236 82*      C      GOOD RUN SO FAR.  READ THE TWO DECODED RECORDS AT END OF TAPE
00237 83*      200 CALL NTRAN(8,22)
00240 84*      CALL NTRAN(8,2,14641,C,L)
00241 85*      210 IF (L+1) 211,210,220
00244 86*      211 IF (L+2) 900,900,220
00247 87*      220 CALL NTRAN(8,2,14641,D,L)
00250 88*      CALL NTRAN(35,1,14641,C,M1)
00251 89*      230 IF (L+1) 231,230,240
00254 90*      231 IF (L+2) 900,900,240
00257 91*      240 CALL NTRAN(35,1,14641,D,M1)
00260 92*      NDX(3,5)=K*14641
00261 93*      NDX(3,6)=(K+1)*14641
00261 94*      C      GOOD RUN.  GO CHAIN IN THE SIMULATION.
00262 95*      CALL CHAIN(2,T)
00262 96*      C      ERROR HAS OCCURRED.  GIVE MESSAGE AND ABORT.
00263 97*      900 WRITE(6,1) LAT,LON
00267 98*      GO TO 930
00270 99*      910 WRITE(6,2) LAT,LON
00274 100*      GO TO 930
00275 101*      920 WRITE(6,3) LAT,LON
00301 102*      930 STOP
00302 103*      1 FORMAT(/' BAD TAPE READ.  LAST READ WAS'2I10)
00303 104*      2 FORMAT(/' BAD DRUM WRITE.  CURRENT IS'2I10)
00304 105*      3 FORMAT(/' EOF ON DRUM.  CURRENT IS'2I10)
00305 106*      END

```

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

```

PHASE 1 TIME = 0 SEC.
PHASE 2 TIME = 0 SEC.
PHASE 3 TIME = 1 SEC.
PHASE 4 TIME = 0 SEC.
PHASE 5 TIME = 0 SEC.
PHASE 6 TIME = 0 SEC.

```

TOTAL COMPILATION TIME = 1 SEC

UNIT FOR WHERSM,WHERSM
UNIVAC 1108 FORTRAN V LEVEL 2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:42

MAIN PROGRAM

STORAGE USED (BLOCK, NAME, LENGTH)

0001	*CODE	003140
0000	*DATA	051716
0002	*BLANK	000004
0003	CB1	013000

EXTERNAL REFERENCES (BLOCK, NAME)

0004	HEIGHT
0005	GRAPH
0006	CLOCKS
0007	HRG
0010	RG
0011	FINDIT
0012	NRDUS
0013	NI01\$
0014	NI02\$
0015	NWDUS
0016	COS
0017	NWBUS
0020	NRBUS
0021	SQRT
0022	NWEF\$
0023	NSTOP\$
0024	NERR3\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0000	051263	1F	0000	051333	10F	0001	001201	1000L	0001	002140	1015G	0001	001302	1020L
0001	002255	1063G	0000	051342	11F	0001	001357	1110L	0001	001375	1120L	0001	001415	1130L
0001	001433	1140L	0001	002530	1177G	0000	051363	12F	0001	001433	1200L	0001	001503	1260L
0001	001511	1290L	0000	051366	13F	0001	002762	1313G	0001	003000	1326G	0000	051474	14F
0001	000031	143G	0000	051507	15F	0000	051511	16F	0000	051535	17F	0001	000105	176G
0000	051566	18F	0000	051275	2F	0000	051573	20F	0001	001514	2010L	0001	001546	2100L
0001	000152	217G	0001	000201	227G	0001	000207	234G	0001	000316	240L	0001	000230	245G
0001	000424	250L	0001	000247	255G	0001	000444	260L	0001	000255	262G	0001	000273	272G
0000	051304	3F	0000	051624	30F	0001	000312	302G	0001	000334	310G	0001	000341	315G
0001	000401	327G	0001	000467	353G	0001	000504	363G	0001	000524	376G	0001	000636	390L
0000	051310	4F	0001	001737	4020L	0001	001761	4040L	0001	000663	422G	0001	000655	425L
0001	000701	430L	0001	000775	454G	0001	000747	460L	0001	001021	467G	0001	000754	470L
0001	001032	476G	0001	001774	4900L	0000	051311	5F	0001	001040	500L	0001	002042	5020L
0001	002051	5040L	0001	002103	5050L	0001	002134	5060L	0001	001053	512G	0001	001056	514G
0001	002127	5200L	0001	002171	5210L	0001	002300	5220L	0001	001100	523G	0001	002553	5236L
0001	002560	5237L	0001	002565	5238L	0001	002571	5239L	0001	002575	5240L	0001	002601	5241L
0001	002605	5242L	0001	002610	5250L	0001	001112	527G	0001	001126	531G	0001	001157	540G
0001	001160	543G	0001	002774	5500L	0001	003031	5550L	0001	001221	556G	0001	003041	5610L

0000 051315 7F 0001 002022 7556 0001 002037 762G 0001 002064 776G 0001 001733 673G
0000 051324 9F 0001 003072 9000L 0000 R 051233 AVASBT 0000 R 051230 AVKTR 0000 051323 8F
0000 004201 A7 0000 R 051157 RND 0000 R 051160 BNDSP 0003 R 011531 CLK 0000 R 051232 AVSUAT
0002 R 000003 CNY 0000 R 004204 CT 0000 R 051161 CYCTM 0000 R 051167 DELTA 0000 R 000002 CNX
0000 R 022252 DY 0000 R 022613 DZ 0003 R 000000 EP 0000 R 051227 ERR 0000 R 021711 DX
0000 R 051146 FXDINI 0000 R 051237 GPKTR 0000 R 051235 GPSUM 0000 R 051236 GPKTR 0003 L 011535 FXD
0000 I 051175 I 0000 I 023154 IFREQ 0000 I 051243 IGP 0000 I 051242 IGT 0000 R 051234 GTSUM
0000 I 051205 IP 0000 I 051177 IPO5 0000 I 051247 IQ0 0000 I 051250 IQ1 0000 I 051155 INR
0000 I 051252 IQ3 0000 I 051253 IQ4 0000 I 051223 IR 0000 I 051251 IQ2 0000 I 051251 IQ2
0000 L 050601 ITIND 0003 I 011530 IU 0000 I 051224 IV 0000 I 000000 IT 0000 I 023160 ITB
0000 I 051143 IS 0000 I 051200 II 0000 I 051202 I2 0000 I 051173 IVMAT 0000 I 051244 IZT
0000 I 051176 J 0000 I 051206 J1 0000 I 051211 J12 0000 I 051204 I3 0000 I 051217 I9
0000 I 051212 K 0000 I 003047 KUF 0000 I 051211 K2 0000 I 051207 J2 0000 I 051210 J3
0000 I 051214 K1CC 0000 I 051152 K2C 0000 I 051144 KZ 0000 I 051222 K1 0000 I 051151 K1C
0000 I 051216 K4CC 0003 I 006116 LOCATR 0000 I 051215 K2CC 0000 I 051153 K3C 0000 I 051154 K4C
0002 I 000001 MXLON 0000 I 051213 NCYC 0000 I 051174 M 0000 I 051170 MAXCYC 0002 I 000000 MNLAT
0003 I 012437 NTYPE 0000 I 051147 NU 0000 I 051172 NITCOL 0000 I 051150 NS 0000 I 051155 NSP
0000 R 051171 RCON 0000 R 051203 REV 0000 R 004177 PSS 0000 R 051145 RADS 0003 P 006460 RANGE
0003 L 007022 S 0000 R 051162 SIMTIM 0003 L 012076 RPT 0001 R 003130 RU 0000 R 051254 RU
0003 R 007363 TL0C 0000 R 011426 TP 0000 R 004207 SP 0000 R 005114 STAT1 0000 R 004212 STAT2
0000 R 051225 T3 0000 R 051226 T4 0000 L 051142 TPOUT 0000 R 051220 T1 0000 R 051231 T2
0000 R 051201 X 0000 R 051163 XMAX 0000 R 012671 VV 0000 R 014475 VV1 0003 R 011167 W
0000 R 051165 YMAX 0000 R 051166 YMIN 0003 R 011533 YNEW 0000 R 051221 Y
0000 R 051240 ZTSUM 0000 R 051245 Z1 0000 R 051246 Z2 0003 R 011534 ZNEW 0000 R 051241 ZPSUM

00101
00103
00104
00105
00106
00107
00110
00111
00111
00112
00114
00116
00121
00125
00127
00127
00131
00131
00157
00160
00161
00161
00204
00205
00206
00207
00210
00211
00212

1*
2*
3*
4*
5*
6*
7*
8*
9*
10*
11*
12*
13*
14*
15*
16*
17*
18*
19*
20*
21*
22*
23*
24*
25*
26*
27*
28*
29*
30*
31*
32*
33*
34*
35*
36*
37*
38*
39*
40*
41*
42*
43*
44*
45*
46*
47*
48*
49*
50*
51*
52*
53*
54*
55*
56*
57*
58*
59*
60*
61*
62*
63*
64*
65*
66*
67*
68*
69*
70*
71*
72*
73*
74*
75*
76*
77*
78*
79*
80*
81*
82*
83*
84*
85*
86*
87*
88*
89*
90*
91*
92*
93*
94*
95*
96*
97*
98*
99*
100*
101*
102*
103*
104*
105*
106*
107*
108*
109*
110*
111*
112*
113*
114*
115*
116*
117*
118*
119*
120*
121*
122*
123*
124*
125*
126*
127*
128*
129*
130*
131*
132*
133*
134*
135*
136*
137*
138*
139*
140*
141*
142*
143*
144*
145*
146*
147*
148*
149*
150*
151*
152*
153*
154*
155*
156*
157*
158*
159*
160*
161*
162*
163*
164*
165*
166*
167*
168*
169*
170*
171*
172*
173*
174*
175*
176*
177*
178*
179*
180*
181*
182*
183*
184*
185*
186*
187*
188*
189*
190*
191*
192*
193*
194*
195*
196*
197*
198*
199*
200*
201*
202*
203*
204*
205*
206*
207*
208*
209*
210*
211*
212*
213*
214*
215*
216*
217*
218*
219*
220*
221*
222*
223*
224*
225*
226*
227*
228*
229*
230*
231*
232*
233*
234*
235*
236*
237*
238*
239*
240*
241*
242*
243*
244*
245*
246*
247*
248*
249*
250*
251*
252*
253*
254*
255*
256*
257*
258*
259*
260*
261*
262*
263*
264*
265*
266*
267*
268*
269*
270*
271*
272*
273*
274*
275*
276*
277*
278*
279*
280*
281*
282*
283*
284*
285*
286*
287*
288*
289*
290*
291*
292*
293*
294*
295*
296*
297*
298*
299*
300*
301*
302*
303*
304*
305*
306*
307*
308*
309*
310*
311*
312*
313*
314*
315*
316*
317*
318*
319*
320*
321*
322*
323*
324*
325*
326*
327*
328*
329*
330*
331*
332*
333*
334*
335*
336*
337*
338*
339*
340*
341*
342*
343*
344*
345*
346*
347*
348*
349*
350*
351*
352*
353*
354*
355*
356*
357*
358*
359*
360*
361*
362*
363*
364*
365*
366*
367*
368*
369*
370*
371*
372*
373*
374*
375*
376*
377*
378*
379*
380*
381*
382*
383*
384*
385*
386*
387*
388*
389*
390*
391*
392*
393*
394*
395*
396*
397*
398*
399*
400*
401*
402*
403*
404*
405*
406*
407*
408*
409*
410*
411*
412*
413*
414*
415*
416*
417*
418*
419*
420*
421*
422*
423*
424*
425*
426*
427*
428*
429*
430*
431*
432*
433*
434*
435*
436*
437*
438*
439*
440*
441*
442*
443*
444*
445*
446*
447*
448*
449*
450*
451*
452*
453*
454*
455*
456*
457*
458*
459*
460*
461*
462*
463*
464*
465*
466*
467*
468*
469*
470*
471*
472*
473*
474*
475*
476*
477*
478*
479*
480*
481*
482*
483*
484*
485*
486*
487*
488*
489*
490*
491*
492*
493*
494*
495*
496*
497*
498*
499*
500*
501*
502*
503*
504*
505*
506*
507*
508*
509*
510*
511*
512*
513*
514*
515*
516*
517*
518*
519*
520*
521*
522*
523*
524*
525*
526*
527*
528*
529*
530*
531*
532*
533*
534*
535*
536*
537*
538*
539*
540*
541*
542*
543*
544*
545*
546*
547*
548*
549*
550*
551*
552*
553*
554*
555*
556*
557*
558*
559*
560*
561*
562*
563*
564*
565*
566*
567*
568*
569*
570*
571*
572*
573*
574*
575*
576*
577*
578*
579*
580*
581*
582*
583*
584*
585*
586*
587*
588*
589*
590*
591*
592*
593*
594*
595*
596*
597*
598*
599*
600*
601*
602*
603*
604*
605*
606*
607*
608*
609*
610*
611*
612*
613*
614*
615*
616*
617*
618*
619*
620*
621*
622*
623*
624*
625*
626*
627*
628*
629*
630*
631*
632*
633*
634*
635*
636*
637*
638*
639*
640*
641*
642*
643*
644*
645*
646*
647*
648*
649*
650*
651*
652*
653*
654*
655*
656*
657*
658*
659*
660*
661*
662*
663*
664*
665*
666*
667*
668*
669*
670*
671*
672*
673*
674*
675*
676*
677*
678*
679*
680*
681*
682*
683*
684*
685*
686*
687*
688*
689*
690*
691*
692*
693*
694*
695*
696*
697*
698*
699*
700*
701*
702*
703*
704*
705*
706*
707*
708*
709*
710*
711*
712*
713*
714*
715*
716*
717*
718*
719*
720*
721*
722*
723*
724*
725*
726*
727*
728*
729*
730*
731*
732*
733*
734*
735*
736*
737*
738*
739*
740*
741*
742*
743*
744*
745*
746*
747*
748*
749*
750*
751*
752*
753*
754*
755*
756*
757*
758*
759*
760*
761*
762*
763*
764*
765*
766*
767*
768*
769*
770*
771*
772*
773*
774*
775*
776*
777*
778*
779*
780*
781*
782*
783*
784*
785*
786*
787*
788*
789*
790*
791*
792*
793*
794*
795*
796*
797*
798*
799*
800*
801*
802*
803*
804*
805*
806*
807*
808*
809*
810*
811*
812*
813*
814*
815*
816*
817*
818*
819*
820*
821*
822*
823*
824*
825*
826*
827*
828*
829*
830*
831*
832*
833*
834*
835*
836*
837*
838*
839*
840*
841*
842*
843*
844*
845*
846*
847*
848*
849*
850*
851*
852*
853*
854*
855*
856*
857*
858*
859*
860*
861*
862*
863*
864*
865*
866*
867*
868*
869*
870*
871*
872*
873*
874*
875*
876*
877*
878*
879*
880*
881*
882*
883*
884*
885*
886*
887*
888*
889*
890*
891*
892*
893*
894*
895*
896*
897*
898*
899*
900*
901*
902*
903*
904*
905*
906*
907*
908*
909*
910*
911*
912*
913*
914*
915*
916*
917*
918*
919*
920*
921*
922*
923*
924*
925*
926*
927*
928*
929*
930*
931*
932*
933*
934*
935*
936*
937*
938*
939*
940*
941*
942*
943*
944*
945*
946*
947*
948*
949*
950*
951*
952*
953*
954*
955*
956*
957*
958*
959*
960*
961*
962*
963*
964*
965*
966*
967*
968*
969*
970*
971*
972*
973*
974*
975*
976*
977*
978*
979*
980*
981*
982*
983*
984*
985*
986*
987*
988*
989*
990*
991*
992*
993*
994*
995*
996*
997*
998*
999*
1000*

PARAMETER L1=225,L2=600
PARAMETER L3=L1/36+1,L4=L1+1,L5=3*L1
DIMENSION IT(L1,L3),KUF(L2),PSS(2),AZ(3),CT(3),SP(3),STAT2(L1,2)
DIMENSION STAT1(L1,10),TP(L1,3),VV(L1,4),VV1(L5,4),DX(L1)
DIMENSION DY(L1),DZ(L1),IFREQ(4),ITB(L1,L3,7),ITIND(L1)
LOGICAL S,FXD,RPT,TPOUT,ITIND
COMMON MNLAT,MXLON,CNX,CNY
COMMON /CB1/EP(L1,14),LOCATR(L4),RANGE(L4),S(L1),TLOC(L1,4),W(L1),
IU,CLK,XNEW,YNEW,ZNEW,FXD(L1),RPT(L1),NTYPE(L1)
DATA TPOUT/,FALSE./
DATA IFREQ/1,6,30,60/
DATA KZ,RADS/3141,57,295779/
DATA FXD,RPT,S/L1*,FALSE.,L1*,FALSE.,L1*,TRUE./
DATA W/L1*,028/
DATA FXDINI/1.0/
C
READ(5,3) NU,NS,K1C,K2C,K3C,K4C,INR,NSP,PSS,BND,BNDSP,
CYCTM,SIMTIM,XMAX,XMIN,YMAX,YMIN
DELTA=CYCTM/FLOAT(NS)
MAXCYC=(60.0*SIMTIM+DELTA)/(FLOAT(NS)*DELTA)
WRITE(6,13) NU,NS,K1C,K2C,K3C,K4C,INR,NSP,SIMTIM,MAXCYC,
PSS(1)=CYCTM/(PSS(1)*60.0)
PSS(2)=CYCTM/(PSS(2)*60.0)
RCON=DELTA/3600.0*1000.0
BND=1000.0*BND
BNDSP=BNDSP*RCON
SIMTIM=60.0*SIMTIM
NITCOL=(NU-1)/36+1


```

00214 31* INPUT THE INITIAL REPORTING UNITS, AND SET UP RPT.
00214 32* M=1
00215 33* READ(5,4) (KUF(I),I=1,INR)
00223 34* WRITE(6,7)
00225 35* WRITE(6,8) (KUF(I),I=1,INR)
00233 36* DO 200 I=1,INR
00236 37* J=KUF(I)
00237 38* 200 RPT(J)=.TRUE.
00237 39* C INPUT THE INITIAL FIXED UNITS, AND SET UP FXD.
00241 40* IF (NSP.EQ.0) GO TO 240
00243 41* READ(5,4) (KUF(I),I=1,NSP)
00251 42* WRITE(6,9)
00253 43* WRITE(6,8) (KUF(I),I=1,NSP)
00261 44* DO 210 I=1,NSP
00264 45* J=KUF(I)
00265 46* W(J)=FXDINI
00266 47* 210 FXD(J)=.TRUE.
00266 48* C INPUT THE UNIT TYPES.
00270 49* READ(5,4) (NTYPE(I),I=1,NU)
00276 50* WRITE(6,18)
00300 51* WRITE(6,8) (NTYPE(I),I=1,NU)
00300 52* C MAKE THE UNIT TO SLOT ASSIGNMENTS.
00306 53* 240 IP0S=0
00307 54* DO 241 I=1,NS
00312 55* 241 KUF(I)=0
00314 56* DO 270 I=NU,1,-1
00314 57* C COMPUTE NUMBER OF SUBCYCLES BETWEEN LOCATION OPERATIONS FOR
00317 58* THIS UNIT.
00317 59* I1=NTYPE(I)
00320 60* I1=IFREQ(I1)
00321 61* X=FLOAT(NS)/FLOAT(I1)
00322 62* I2=X
00323 63* X=X-FLOAT(I2)
00324 64* REM=0.0
00325 65* EP(I,14)=FLOAT(NS)/(5.*FLOAT(I1))
00326 66* DO 260 I3=1,I1
00331 67* REM=REM+X
00331 68* C FIND SLOT TO ASSIGN. IF SLOT FIRST TRIED IS NOT AVAILABLE,
00331 69* C TRY ADJACENT SLOTS UNTIL AN AVAILABLE ONE IS FOUND.
00332 70* IP0S=IP0S+I2+INT(REM)
00333 71* IF (IP0S.GT.NS) IP0S=IP0S-NS
00335 72* IF (KUF(IP0S).EQ.0) GO TO 260
00337 73* IP0S=IP0S+1
00340 74* IF (IP0S.GT.NS) IP0S=IP0S-NS
00342 75* GO TO 250
00343 76* 260 KUF(IP0S)=I
00345 77* 270 CONTINUE
00347 78* WRITE(6,10)
00351 79* WRITE(6,8) (KUF(I),I=1,NS)
00351 80* C INPUT MOVEMENT DATA AND STARTING POSITIONS.
00357 81* WRITE(6,11)
00361 82* IP=1
00362 83* DO 500 I=1,NU
00365 84* READ(5,30) IU,I1,I2,I3,J1,J2,J3,(CT(J),AZ(J),SP(J),J=1,3)
00365 85* C CONVERT DEGREES,MINUTES, AND SECONDS TO SECONDS.
00404 86* I1=(I1*60+I2)*60+I3
00405 87* J1=(J1*60+J2)*60+J3

```



```
88* J405 C RT NDS MEI
89* TP(IU,1)=(MXLON*60-J177/3.0*CNY)
90* TP(IU,2)=(I1-MNLAT*60)/(3.0*CNY)
91* IF (NTYPE(IU).LT.3) GO TO 390
92* AIRCRAFT HAVE A RANDOM ALTITUDE ASSIGNED TO THEM.
93* IF (NTYPE(IU).EQ.3) TP(IU,3)=100.+5000.*RU(KZ)
94* IF (NTYPE(IU).EQ.4) TP(IU,3)=100.+4000.*RU(KZ)
95* GO TO 425
96* INITIATE HEIGHT RETRIEVAL.
97* 390 CALL HEIGHT(-1,-1,TP(IU,1),TP(IU,2),TP(IU,3),M)
98* CONVERT MOVEMENT DATA INTO DIRECTION COSINES AND BUILD
99* THE VELOCITY VECTORS.
100* 425 VV(IU,4)=IP
101* DO 450 J=1,3
102* IF (CT(J).LE.0.0) GO TO 460
103* CT(J)=3600.*CT(J)
104* IF (J.EQ.1) GO TO 430
105* CT(J)=CT(J)+CT(J-1)
106* VV1(IP,1)=COS((AZ(J)-90.0)/RADS)*SP(J)*RCON
107* VV1(IP,2)=COS(AZ(J)/RADS)*SP(J)*RCON
108* VV1(IP,4)=IP+1
109* VV1(IP,3)=CT(J)
110* IP=IP+1
111* IF (CT(J).GT.SIMTIM) GO TO 460
112* 450 CONTINUE
113* J=3
114* 460 VV1(IP-1,4)=-5.0
115* J12=J
116* WAIT FOR END OF HEIGHT RETRIEVAL.
117* 470 IF (M.EQ.-1) GO TO 470
118* WRITE(6,12) IU,NTYPE(IU),(TP(IU,J),J=1,3),AZ(1),SP(1),CT(1)
119* IF (J12.EQ.1) GO TO 500
120* WRITE(6,6) (AZ(J),SP(J),CT(J),J=2,J12)
121* INITIALIZE EP, THE ESTIMATED POSITIONS MATRIX.
122* DO 480 J=1,3
123* EP(IU,J)=TP(IU,J)
124* EP(IU,J+3)=TP(IU,J)
125* EP(IU,J+6)=TP(IU,J)
126* EP(IU,J+9)=TP(IU,J)
127* 480 EP(IU,J+9)=TP(IU,J)
128* 500 CONTINUE
129* IF (TPOUT) WRITE(9) ((TP(I,J),J=1,3),NTYPE(I),I=1,NU)
130* INPUT THE INTERVISIBILITY TABLES.
131* DO 520 K=1,7
132* 520 READ(7) ((ITB(I,J,K),J=1,NITCOL),I=1,NU)
133* INITIALIZE IT, THE CURRENT INTERVISIBILITY TABLE.
134* DO 530 I=1,NU
135* DO 530 J=1,NITCOL
136* IT(I,J)=ITR(I,J,1)
137* CALL GRAPH(NU,TP(1,1),TP(1,2),TRUE,XMAX,XMIN,YMAX,YMIN)
138* * * * BEGIN THE PROCESSING OF A CYCLE. * * *
139* C
140* C
141* 1000 NCYC=NCYC+1
142* K1CC=K1CC+1
143* K2CC=K2CC+1
144* K4CC=K4CC+1
145* DO 4900 I9=1,NS
146* CALL CLOCKS(T1)
```



```

88*
00405
00406
00407
00410
00411
00412
00414
00416
00417
00418
00419
00420
00421
00424
00426
00427
00431
00432
00433
00434
00435
00436
00437
00441
00443
00444
00445
00446
00450
00463
00465
00466
00475
00500
00501
00502
00503
00505
00507
00507
00522
00525
00525
00537
00542
00545
00550
00550
00550
00550
00550
00551
00552
00553
00554
00555
00560

C
TP(IU,1)=(MXLON*60-J17/3.0*CNV)
TP(IU,2)=(I1-MNLAT*60)/(3.0*CNV)
IF (NTYPE(IU).LT.3) GO TO 390
AIRCRFT HAVE A RANDOM ALTITUDE ASSIGNED TO THEM.
IF (NTYPE(IU).EQ.3) TP(IU,3)=100.+5000.*RU(KZ)
IF (NTYPE(IU).EQ.4) TP(IU,3)=100.+4000.*RU(KZ)
GO TO 425
INITIATE HEIGHT RETRIEVAL.
390 CALL HEIGHT(-1.,-1.,TP(IU,1),TP(IU,2),TP(IU,3),M)
CONVERT MOVEMENT DATA INTO DIRECTION COSINES AND BUILD
THE VELOCITY VECTORS.
425 VV(IU,4)=IP
DO 450 J=1,3
IF (CT(J).LE.0.0) GO TO 460
CT(J)=3600.*CT(J)
IF (J.EQ.1) GO TO 430
CT(J)=CT(J)+CT(J-1)
430 VV1(IP,1)=COS((AZ(J)-90.0)/RADS)*SP(J)*RCON
VV1(IP,2)=COS(AZ(J)/RADS)*SP(J)*RCON
VV1(IP,4)=IP+1
VV1(IP,3)=CT(J)
IP=IP+1
IF (CT(J).GT.SIMTIM) GO TO 460
450 CONTINUE
J=3
460 VV1(IP-1,4)=-5.0
J12=J
WAIT FOR END OF HEIGHT RETRIEVAL.
470 IF (M.EQ.-1) GO TO 470
WRITE(6,12) IU,NTYPE(IU),(TP(IU,J),J=1,3),AZ(1),SP(1),CT(1)
IF (J12.EQ.1) GO TO 500
WRITE(6,6) (AZ(J),SP(J),CT(J),J=2,J12)
INITIALIZE EP, THE ESTIMATED POSITIONS MATRIX.
DO 480 J=1,3
EP(IU,J)=TP(IU,J)
EP(IU,J+3)=TP(IU,J)
EP(IU,J+6)=TP(IU,J)
480 EP(IU,J+9)=TP(IU,J)
500 CONTINUE
IF (IPOUT) WRITE(9) ((TP(I,J),J=1,3),NTYPE(I),I=1,NU)
INPUT THE INTERVISIBILITY TABLES.
DO 520 K=1,7
520 READ(7) ((ITB(I,J,K),J=1,NITCOL),I=1,NU)
INITIALIZE IT, THE CURRENT INTERVISIBILITY TABLE.
DO 530 I=1,NU
DO 530 J=1,NITCOL
530 IT(I,J)=ITR(I,J,1)
CALL GRAPH(NU,TP(1,1),TP(1,2),TRUE,XMAX,XMIN,YMAX,YMIN)
** * BEGIN THE PROCESSING OF A CYCLE. * * *
1000 NCYC=NCYC+1
K1CC=K1CC+1
K2CC=K2CC+1
K4CC=K4CC+1
DO 4900 I9=1,NS
CALL CLOCKS(I1)
145*

```



```

00361 146* CLK=CLK+DELTA
00362 147* IU=KUF(I9)
00363 148* BND=BND+BNDSP
00364 149* C DETERMINE THE LOCATEE'S NEW POSITION BY INCREMENTING THE
00365 150* OLD POSITION WITH VV. DO NOT MOVE A FIXED UNIT.
00366 151* IF (FXD(IU)) GO TO 1200
00367 152* IF (VV(IU,3).GT.CLK) GO TO 1020
00368 153* C CURRENT VV IS OUTDATED. GET A NEW ONE FROM VV1,
00369 154* IF AVAILABLE.
00370 155* IF (VV(IU,4).LT.0.0) GO TO 1200.
00371 156* X=TP(IU,1)
00372 157* Y=TP(IU,2)
00373 158* K1=VV(IU,4)+.5
00374 159* DO 1010 J=1,4
00375 160* VV(IU,J)=VV1(K1,J)
00376 161* 1010 TP(IU,1)=TP(IU,1)+VV(IU,1)
00377 162* TP(IU,2)=TP(IU,2)+VV(IU,2)
00378 163* C GET THE LOCATEE'S HEIGHT.
00379 164* IF (NTYPE(IU).LT.3) CALL HEIGHT(X,Y,TP(IU,1),TP(IU,3),M)
00380 165* IF LOCATEE IS BEYOND THE BOUNDARY, CHANGE HIS DIRECTION.
00381 166* 1050 IF (TP(IU,1).LT.BND) GO TO 1110
00382 167* IF (VV(IU,1).GT.0.0) VV(IU,1)=VV(IU,1)
00383 168* GO TO 1120
00384 169* 1110 IF (TP(IU,1).GT.0.0) GO TO 1120
00385 170* IF (VV(IU,1).LT.0.0) VV(IU,1)=VV(IU,1)
00386 171* 1120 IF (TP(IU,2).LT.BND) GO TO 1130
00387 172* IF (VV(IU,2).GT.0.0) VV(IU,2)=VV(IU,2)
00388 173* GO TO 1140
00389 174* 1130 IF (TP(IU,2).GT.0.0) GO TO 1140
00390 175* IF (VV(IU,2).LT.0.0) VV(IU,2)=VV(IU,2)
00391 176* 1140 CONTINUE
00392 177* C DETERMINE ALL OTHER UNITS' NEW POSITIONS. PROCEDURE IS THE
00393 178* SAME HERE AS FOR THE LOCATEE, BUT NO HEIGHT RETRIEVAL AND
00394 179* NO BOUNDARY CHECKING.
00395 180* 1200 DO 1290 I=1,NU
00396 181* IF (FXD(I)) GO TO 1290
00397 182* IF (I.EQ.IU) GO TO 1290
00398 183* IF (VV(I,3).GT.CLK) GO TO 1260
00399 184* IF (VV(I,4).LT.0.0) GO TO 1290
00400 185* K1=VV(I,4)+.5
00401 186* DO 1230 J=1,4
00402 187* VV(I,J)=VV1(K1,J)
00403 188* 1260 TP(I,1)=TP(I,1)+VV(I,1)
00404 189* TP(I,2)=TP(I,2)+VV(I,2)
00405 190* 1290 CONTINUE
00406 191* 2000 IR=0
00407 192* C WAIT FOR END OF HEIGHT RETRIEVAL.
00408 193* 2010 IF (M.EQ.-1) GO TO 2010
00409 194* C PERTURB HEIGHT FOR NON-FIXED UNITS, AND STORE IN EP.
00410 195* CALL HRG(X)
00411 196* IF (FXD(IU)) X=0.0
00412 197* X=X+TP(IU,3)
00413 198* EP(IU,6)=X
00414 199* EP(IU,9)=X
00415 200* EP(IU,12)=X
00416 201* IF (FXD(IU)) GO TO 4900
00417 202* C COMPUTE RANGES FOR THIS SUBCYCLE.
00418 203* DO 2100 J=1,NU

```



```

00672 204* C IF LOCATOR CANDIDATE IS NOT A REPORTER, SKIP THAT RANGE
00672 205* C CALCULATION.
00675 206* IF (.NOT.RPT(J)) GO TO 2100
00675 207* IF (J.EQ.IU) GO TO 2100
00677 208* C RETRIEVE INTERVISIBILITY FOR LOCATEE-LOCATOR PAIR.
00701 209* J1=(J-1)/36
00702 210* IV=FLD(J-J1*36,1,IT(IU,J1+1))
00702 211* C IF NO LOS, SKIP RANGE CALCULATION.
00703 212* IF (IV.EQ.0) GO TO 2100
00705 213* IR=IR+1
00705 214* C COMPUTE RANGE AND PERTURB IT.
00706 215* RANGE(IR)=SORT((TP(J,1)-TP(IU,1))**2+(TP(J,2)-TP(IU,2))**2
* +(TP(J,3)-TP(IU,3))**2)
00706 216* CALL RG(X)
00707 217* RANGE(IR)=RANGE(IR)+X
00710 218* LOCATR(IR)=J
00711 219*
00712 220* C 2100 CONTINUE
00714 221* RANGE(IR+1)=-5.0
00714 222* C GO ATTEMPT TO LOCATE THE LOCATEE.
00715 223* 3000 CALL CLOCKS(T3)
00716 224* CALL FINDIT
00717 225* CALL CLOCKS(T4)
00717 226* C ACCUMULATE ERROR STATISTICS ON THE LOCATEE.
00720 227* 4000 IF (.NOT.S(IU)) GO TO 4000
00723 228* DX(IU)=EP(IU,4)-TP(IU,1)
00723 229* DY(IU)=EP(IU,5)-TP(IU,2)
00724 230* DZ(IU)=EP(IU,6)-TP(IU,3)
00725 231* ERR=SORT(DX(IU)**2+DY(IU)**2)
00726 232* STAT1(IU,4)=STAT1(IU,4)+ERR
00727 233* STAT2(IU,1)=STAT2(IU,1)+1.0
00730 234* STAT2(IU,2)=STAT2(IU,2)+1.0
00731 235* AVKTR=AVKTR+1.0
00732 236* IF (STAT1(IU,5)-ERR) 4010,4020,4020
00735 237* STAT1(IU,5)=ERR
00736 238* STAT1(IU,6)=NCYC
00737 239* ERR=ABS(DZ(IU))
00740 240* STAT1(IU,10)=STAT1(IU,10)+ERR
00741 241* IF (STAT1(IU,8).GE.ERR) GO TO 4040
00743 242* STAT1(IU,8)=ERR
00744 243* STAT1(IU,9)=NCYC
00745 244* CALL CLOCKS(T2)
00746 245* AVSUBT=AVSUBT+T2-T1
00747 246* AVASBT=AVASBT+T4-T3
00747 247*
00747 248* C * * * END OF THE PROCESSING FOR A CYCLE. * * *
00747 249* C
00750 250* C 4900 CONTINUE
00750 251* C UPDATE THE INTERVISIBILITY TABLE.
00752 252* 5000 IF (CLK.LT.FLOAT(1200*(IVMAT-1))) GO TO 5040
00752 253* C IF THE TIME IS HIGH, COMPLETE THE OVERLAYING OF THE CURRENT
00752 254* C INTERVISIBILITY MATRIX, AND ARRANGE FOR FUTURE USE OF THE
00752 255* C NEXT INTERVISIBILITY MATRIX IN SEQUENCE.
00754 256* DO 5020 I=1,NU
00757 257* IF (ITIND(I)) GO TO 5020
00761 258* DO 5010 J=1,NITCOL
00764 259* IT(I,J)=ITR(I,J,IVMAT)
00766 260* ITIND(I)=.FALSE.
00770 261* IVMAT=IVMAT+1

```



```

00771
263* 5040 IF (K2CC.LT.K2C) GO TO 5200
264* K2CC=0
265* UPDATE THE CURRENT INTERVISIBILITY MATRIX BY RANDOMLY
266* FINDING K3C ROWS THAT HAVE NOT BEEN CHANGED AND
267* CHANGING THEM.
268* DO 5090 I=1,K3C
269* I1=FLOAT(NU)*RU(KZ)+1.0
270* I2=I1
271* IF (.NOT.ITIND(I2)) GO TO 5060
272* I2=I2+1
273* IF (I2.GT.NU) I2=I2-NU
274* IF (I2.NE.I1) GO TO 5050
275* GO TO 5200
276* 5060 DO 5070 J=1,NIICOL
277* 5070 IT(I2,J)=ITB(I2,J,IVMAT)
278* 5090 ITIND(I2)=.TRUE.
279* PREPARE AND PRINT INTERMEDIATE ERROR OUTPUT.
280* 5200 IF (NCYC.EQ.MAXCYC) GO TO 5210
281* IF (K1CC.LT.K1C) GO TO 5500
282* 5210 GTSUM=0.0
283* GPSUM=0.0
284* GTKTR=0.0
285* GPKTR=0.0
286* ZTSUM=0.0
287* ZPSUM=0.0
288* IGT=1
289* IGP=1
290* IZT=1
291* AVSUBT=AVSUBT/AVKTR
292* AVASBT=AVASBT/AVKTR
293* WRITE HEADINGS.
294* WRITE(6,20) NCYC,CLK,AVSUBT,AVASBT
295* WRITE(6,16) NCYC,K1CC
296* WRITE(6,17)
297* IF (TPOUT) WRITE(9) NCYC,CLK,K1CC
298* PREPARE AND PRODUCE THE BODY OF THE OUTPUT TABLE.
299* ONE LINE AT A TIME.
300* DO 5250 I=1,NU
301* STAT1(I,1)=STAT1(I,1)+STAT1(I,4)
302* STAT1(I,7)=STAT1(I,7)+STAT1(I,10)
303* IF (STAT1(I,2).GE.STAT1(I,5)) GO TO 5220
304* STAT1(I,2)=STAT1(I,5)
305* STAT1(I,3)=STAT1(I,6)
306* ACCUMULATE STATISTICS FOR LAST LINE OF TABLE.
307* 5220 IF (STAT1(I,2).GT.STAT1(IGT,2)) IGT=I
308* IF (STAT1(I,5).GT.STAT1(IGP,5)) IGP=I
309* IF (STAT1(I,8).GT.STAT1(IZT,8)) IZT=I
310* GTSUM=GTSUM+STAT1(I,1)
311* GPSUM=GPSUM+STAT1(I,4)
312* ZTSUM=ZTSUM+STAT1(I,7)
313* ZPSUM=ZPSUM+STAT1(I,10)
314* GTKTR=GTKTR+STAT2(I,1)
315* GPKTR=GPKTR+STAT2(I,2)
316* J1=STAT1(I,3)+.5
317* J2=STAT1(I,6)+.5
318* J3=STAT1(I,9)+.5
319* X=0.0
01112

```


01113 IF (STAT2(I,1),GT,0.0) X=STAT1(I,1)/STAT2(I,1)
320* Y=0.0
01115 IF (STAT2(I,2),GT,0.0) Y=STAT1(I,4)/STAT2(I,2)
321* Z1=0.0
01116 IF (STAT2(I,1),GT,0.0) Z1=STAT1(I,7)/STAT2(I,1)
322* Z2=0.0
01121 IF (STAT2(I,2),GT,0.0) Z2=STAT1(I,10)/STAT2(I,2)
323* IF (STAT2(I,2),GT,0.0) Z2=STAT1(I,10)/STAT2(I,2)
01124 WRITE OUT A LINE OF THE TABLE.
324* WRITE(6,1) I,X,Z1,STAT1(I,2),J1,STAT1(I,8),J3,Y,Z2,STAT1(I,5),J2,
325* DX(I),DY(I),DZ(I),S(I),RPT(I),FXD(I),W(I)
01126 IF (TPOUT) WRITE(9) I,X,Z1,STAT1(I,2),J1,STAT1(I,8),J3,Y,Z2,
326* STAT1(I,5),J2,DX(I),DY(I),DZ(I),S(I),RPT(I),FXD(I),
01127 W(I),(TP(I,J),J=1,3)
327* *
01128 *
01129 *
01130 *
01131 *
01132 *
01133 *
01134 *
01135 *
01136 *
01137 *
01138 *
01139 *
01140 *
01141 *
01142 *
01143 *
01144 *
01145 *
01146 *
01147 *
01148 *
01149 *
01150 *
01151 *
01152 *
01153 *
01154 *
01155 *
01156 *
01157 *
01158 *
01159 *
01160 *
01161 *
01162 *
01163 *
01164 *
01165 *
01166 *
01167 *
01168 *
01169 *
01170 *
01171 *
01172 *
01173 *
01174 *
01175 *
01176 *
01177 *
01178 *
01179 *
01180 *
01181 *
01182 *
01183 *
01184 *
01185 *
01186 *
01187 *
01188 *
01189 *
01190 *
01191 *
01192 *
01193 *
01194 *
01195 *
01196 *
01197 *
01198 *
01199 *
01200 *
01201 *
01202 *
01203 *
01204 *
01205 *
01206 *
01207 *
01208 *
01209 *
01210 *
01211 *
01212 *
01213 *
01214 *
01215 *
01216 *
01217 *
01218 *
01219 *
01220 *
01221 *
01222 *
01223 *
01224 *
01225 *
01226 *
01227 *
01228 *
01229 *
01230 *
01231 *
01232 *
01233 *
01234 *
01235 *
01236 *
01237 *
01238 *
01239 *
01240 *
01241 *
01242 *
01243 *
01244 *
01245 *
01246 *
01247 *
01248 *
01249 *
01250 *
01251 *
01252 *
01253 *
01254 *
01255 *
01256 *
01257 *
01258 *
01259 *
01260 *
01261 *
01262 *
01263 *
01264 *
01265 *
01266 *
01267 *
01268 *
01269 *
01270 *
01271 *
01272 *
01273 *
01274 *
01275 *
01276 *
01277 *
01278 *
01279 *
01280 *
01281 *
01282 *
01283 *
01284 *
01285 *
01286 *
01287 *
01288 *
01289 *
01290 *
01291 *
01292 *
01293 *
01294 *
01295 *
01296 *
01297 *
01298 *
01299 *
01300 *
01301 *
01302 *
01303 *
01304 *
01305 *
01306 *
01307 *
01308 *
01309 *
01310 *
01311 *
01312 *
01313 *
01314 *
01315 *
01316 *
01317 *
01318 *
01319 *
01320 *
01321 *
01322 *
01323 *
01324 *
01325 *
01326 *
01327 *
01328 *
01329 *
01330 *
01331 *
01332 *
01333 *
01334 *
01335 *
01336 *
01337 *
01338 *
01339 *
01340 *
01341 *
01342 *
01343 *
01344 *
01345 *
01346 *
01347 *
01348 *
01349 *
01350 *
01351 *
01352 *
01353 *
01354 *
01355 *
01356 *
01357 *
01358 *
01359 *
01360 *
01361 *
01362 *
01363 *
01364 *
01365 *
01366 *
01367 *
01368 *
01369 *
01370 *
01371 *
01372 *
01373 *
01374 *
01375 *
01376 *
01377 *
01378 *
01379 *
01380 *
01381 *
01382 *
01383 *
01384 *
01385 *
01386 *
01387 *
01388 *
01389 *
01390 *
01391 *
01392 *
01393 *
01394 *
01395 *
01396 *
01397 *
01398 *
01399 *
01400 *
01401 *
01402 *
01403 *
01404 *
01405 *
01406 *
01407 *
01408 *
01409 *
01410 *
01411 *
01412 *
01413 *
01414 *
01415 *
01416 *
01417 *
01418 *
01419 *
01420 *
01421 *
01422 *
01423 *
01424 *
01425 *
01426 *
01427 *
01428 *
01429 *
01430 *
01431 *
01432 *
01433 *
01434 *
01435 *
01436 *
01437 *
01438 *
01439 *
01440 *
01441 *
01442 *
01443 *
01444 *
01445 *
01446 *
01447 *
01448 *
01449 *
01450 *
01451 *
01452 *
01453 *
01454 *
01455 *
01456 *
01457 *
01458 *
01459 *
01460 *
01461 *
01462 *
01463 *
01464 *
01465 *
01466 *
01467 *
01468 *
01469 *
01470 *
01471 *
01472 *
01473 *
01474 *
01475 *
01476 *
01477 *
01478 *
01479 *
01480 *
01481 *
01482 *
01483 *
01484 *
01485 *
01486 *
01487 *
01488 *
01489 *
01490 *
01491 *
01492 *
01493 *
01494 *
01495 *
01496 *
01497 *
01498 *
01499 *
01500 *


```

378* IF (NCYC.EQ.MAXCYC) GO TO 5610
379*   UPDATE THE STATIONARY UNIT LIST.
380*   5500 DO 5550 I=1,NU
381*   C   AIRCRAFT CANNOT BE STATIONARY.
382*   IF (NTYPE(I).GT.2) GO TO 5550
383*   J=1
384*   IF (FXD(I)) J=2
385*   IF (RU(KZ).GT.PSS(J)) GO TO 5550
386*   FXD(I)=.NOT.FXD(I)
387*   W(I)=0.0
388*   5550 CONTINUE
389*   C   PRODUCE THE INTERMEDIATE XY PLOTS.
390*   5600 IF (K4CC.LT.K4C) GO TO 1000
391*   K4CC=0
392*   5610 CALL GRAPH(NU,TP(1,1),TP(1,2),TRUE,XMAX,XMIN,YMAX,YMIN)
393*   CALL GRAPH(NU,EP(1,7),EP(1,8),EST,XMAX,XMIN,YMAX,YMIN)
394*   IF (NCYC.EQ.MAXCYC) GO TO 9000
395*   GO TO 1000
396*   9000 IF (TPOUT) ENDFILE 9
397*   STOP
398*   C   FORMAT STATEMENTS AND INTERNAL SUBPROGRAMS FOLLOW.
399*   1 FORMAT(1X14,3E10.5,I4,E10.5,I4,2X,3E9.3,L2,2L1,E10.5)
400*   2 FORMAT(/, ALL'3E10.5,I4,E10.5,I4,2X3E10.5,I4)
401*   3 FORMAT(6I5,5F10.5/6F10.5)
402*   4 FORMAT(20I4)
403*   5 FORMAT(15,F5.0,I5)
404*   6 FORMAT(53X3E14.8)
405*   7 FORMAT(/, INITIAL REPORTING UNITS ARE:))
406*   8 FORMAT(26I5)
407*   9 FORMAT(/, INITIAL STATIONARY UNITS ARE:))
408*   10 FORMAT(/, ASSIGNMENT OF UNITS TO SLOTS IS:))
409*   11 FORMAT(/, UNIT TYPE STARTING X STARTING Y STARTING Z'6X'A
410*   *ZIMUTH'8X'SPEED'6X'CHANGE TIME')
411*   12 FORMAT(15,I6,6E14.8)
412*   13 FORMAT(/, NUMBER OF UNITS =I9/, IV UPDATE COUNTER =I9/, ROWS IN IV UPDATE =I9/
413*   * COUNTER =I9/, XY PLOT COUNTER =I9/
414*   * MAX NUMBER RANGES =I9/, NUM. STAT. UNITS =I9/, SIMULATION T
415*   *IME =F9.2/, MAX NUMBER CYCLES =I9/, SUBCYCLE DELTA T =F9.4/, A
416*   *VG TIME MOVING =F9.2/, AVG TIME STOPPED =F9.2/, UPPER BOUNDA
417*   *RIES =F9.4/, BOUNDARY SPEED =F9.2)
418*   14 FORMAT(/53X'QUADRANT COUNTS OF ERRORS'//60X13,5X13/64X13/60X13,
419*   *5X13)
420*   15 FORMAT(21(1X5I1))
421*   16 FORMAT(/14X'ERROR STATS. OVER ALL'I4,' CYCLES'11X'ERROR STATS. OVE
422*   *R LAST'I3,' CYCLES'4X'CURRENT COORDINATE ERRORS')
423*   17 FORMAT(/, UNIT XY AVG'5X'Z AVG'4X'XY MAX CYC Z MAX CYC'4X'XY
424*   * AVG'5X'Z AVG'4X'XY MAX CYC'5X'EX-TX'4X'EY-TY'4X'EZ-TZ SRF WEI
425*   *GHT'//)
426*   18 FORMAT(/, THE UNIT TYPES ARE:))
427*   20 FORMAT(/, CYCLE NUMBER,I5,, TIME =F10.3,' S. AVERAGE SIMULAT
428*   *ION SUBCYCLE TIME =F7.4,' S. AVERAGE ALGORITHM SUBCYCLE TIME =F
429*   *7.4,' S.)
430*   30 FORMAT(13,18X2(I3,2I2),3(1X3F4.0))
431*   FUNCTION RU(KZ)
432*   KZ=KZ*2045
433*   KZ=KZ-(KZ/4194304)*4194304
434*   RU=KZ
435*
01323
01323
01325
01325
01325
01330
01332
01333
01333
01335
01337
01340
01341
01341
01343
01345
01346
01347
01350
01352
01352
01353
01355
01355
01356
01357
01360
01361
01362
01363
01364
01364
01365
01366
01367
01370
01370
01371
01372
01372
01372
01372
01372
01373
01373
01373
01374
01375
01375
01375
01376
01376
01376
01376
01377
01377
01400
01400
01400
01400
01401
01401
01402
01402
01406
01407

```


01410 436* RU=RU*2.3841858E-7
01411 437* RETURN
01412 438* END

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

PHASE 1 TIME = 1 SEC.
PHASE 2 TIME = 0 SEC.
PHASE 3 TIME = 2 SEC.
PHASE 4 TIME = 0 SEC.
PHASE 5 TIME = 1 SEC.
PHASE 6 TIME = 1 SEC.

TOTAL COMPILATION TIME = 5 SEC

QUIT FOR HEIGHT, HEIGHT
UNIVAC 1108 FORTRAN V LEVEL 2205 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:48

SUBROUTINE HEIGHT ENTRY POINT 000270

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000306
0000 *DATA 000054
0002 *BLANK 000073

EXTERNAL REFERENCES (BLOCK, NAME)

0003 NTRAN
0004 NWDU\$
0005 NIO2\$
0006 NSTOP\$
0007 NERR3\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0000	000011	IF	0001	000057	100L	0001	000144	105L	0001	000161	110L	0001	000164	120L
0001	000167	130L	0001	000173	200L	0002	R	000002	CNY	0002	000003	CNY	0000	I 000004 I
0000	I	000002	IFX	0000	I	000003	IFY	0000	I	000000	IPX	0000	I	000006 I1
0000	I	000005	J	0000	I	000007	J1	0002	I	000000	MNLAT	0000	I	000010 N
0002	I	000004	NDX											

00101	1*	SUBROUTINE HEIGHT(PX, PY, FX, FY, F7, M)
00103	2*	PARAMETER L6=5, L7=11
00104	3*	COMMON MNLAT, MXLON, CNX, CNY, NDX(L6, L7)
00105	4*	DATA CNX, CNY/.0145515310, .0107798475/
00105	5*	C CONVERT FROM POSITION IN METERS TO POSITION IN TERMS OF
00105	6*	C THREE SECONDS OF LATITUDE AND LONGITUDE.
00110	7*	IPX=PX*CNX+.5
00111	8*	IPY=PY*CNX+.5
00112	9*	IFX=FX*CNX+.5
00113	10*	IFY=FY*CNX+.5
00114	11*	IF (IPX.NE.IFX) GO TO 100
00116	12*	IF (IPY.NE.IFY) GO TO 100
00116	13*	C IF FUTURE POSITION IS ON SAME TERRAIN PLATFORM AS PRESENT
00116	14*	C POSITION, FEED BACK SAME HEIGHT.
00120	15*	M=1
00121	16*	RETURN
00121	17*	C COMPUTE THE SUBSCRIPTS FOR THE REQUIRED TERRAIN BLOCK.
00122	18*	100 I=IFY/120+1
00123	19*	J=IFX/120+1
00124	20*	IF (NDX(I, J).GE.0) GO TO 200
00124	21*	C IF THAT TERRAIN BLOCK DOES NOT EXIST, UNIT MAY BE ON A
00124	22*	C BOUNDARY BETWEEN TWO OR FOUR BLOCKS. CHECK EXISTENCE OF


```

00124      23*      C      THOSE BLOCKS.
00126      24*      I1=(IFY-1)/120+1
00127      25*      IF (NDX(I1,J).GE.0) GO TO 110
00131      26*      J1=(IFX-1)/120+1
00132      27*      IF (NDX(I,J1).GE.0) GO TO 120
00134      28*      IF (NDX(I1,J1).GE.0) GO TO 130
00136      29*      105 WRITE(6,1) FX,FY,I,J,IFX,IFY
00146      30*      1 FORMAT(/, ' NO HEIGHT FOR'2E14.8,' NEED BLOCK'2I5,' REDUCED COORDINA
00146      31*      *TES ARE:'2I5)
00147      32*      STOP
00150      33*      110 I=I1
00151      34*      GO TO 200
00152      35*      120 J=J1
00153      36*      GO TO 200
00154      37*      130 I=I1
00155      38*      J=J1
00155      39*      C      CHECK VALIDITY OF TERRAIN BLOCK SUBSCRIPTS.
00156      40*      200 IF (I.GT.L6) GO TO 105
00160      41*      IF (I.LT.1) GO TO 105
00162      42*      IF (J.GT.L7) GO TO 105
00164      43*      IF (J.LT.1) GO TO 105
00164      44*      C      COMPUTE SUBSCRIPTS FOR LOCATION OF HEIGHT WITHIN THE BLOCK.
00166      45*      I1=IFY-120*(I-1)+1
00167      46*      J1=IFX-120*(J-1)+1
00167      47*      C      COMPUTE POSITION OF REQUIRED HEIGHT ON THE DRUM.
00170      48*      N=NDX(I,J)+121*(J1-1)+I1-1
00170      49*      C      REWIND THE DRUM, POSITION TO REQUIRED HEIGHT, AND READ IT N.
00171      50*      CALL NTRAN(35,10,6,N,2,1,FZ,M)
00172      51*      RETURN
00173      52*      END

```

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

```

PHASE 1 TIME = 0 SEC.
PHASE 2 TIME = 0 SEC.
PHASE 3 TIME = 0 SEC.
PHASE 4 TIME = 1 SEC.
PHASE 5 TIME = 0 SEC.
PHASE 6 TIME = 0 SEC.

```

TOTAL COMPILATION TIME = 1 SEC

SUBROUTINE HRG ENTRY POINT 000072

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000077
0000 *DATA 000021
0002 *BLANK 000000

EXTERNAL REFERENCES (BLOCK, NAME)

0003 HRGU
0004 ALOG
0005 SQRT
0006 COS
0007 NERR3\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0001	000057	50L	0004	R	000000	ALOG	0000	R	000005	A1	0000	R	000004	A1S	0000	R	000006	A2
0000	I	000001	IFLAG	0000	R	000000	PI	0000	R	000002	U1	0000	R	000003	U2			

00101 1* SUBROUTINE HRG(X)
00103 2* DATA PI/3.14159265358979/
00105 3* DATA IFLAG/0/
00107 4* IF (IFLAG.GT.0) GO TO 50
00111 5* CALL HRGU(U1,U2)
00112 6* A1S=-2.0*ALOG(U1)
00113 7* A1=SQRT(A1S)
00114 8* A2=2.0*PI*U2
00115 9* U1=A1*COS(A2)
00116 10* U2=SQRT(A1S-U1**2)
00117 11* IF (A2.GT.PI) U2=-U2
00121 12* X=U1*14.0
00122 13* IFLAG=1
00123 14* RETURN
00124 15* 50 X=U2*14.0
00125 16* IFLAG=0
00126 17* RETURN
00127 18* END

END OF UNIVAC 1108 FORTRAN V COMPILATION. 0 *DIAGNOSTIC* MESSAGE(S)

PHASE 1	TIME =	0 SEC.
PHASE 2	TIME =	0 SEC.
PHASE 3	TIME =	0 SEC.
PHASE 4	TIME =	0 SEC.
PHASE 5	TIME =	1 SEC.

PHASE 6 TIME = 0 SEC.

TOTAL COMPILATION TIME = 1 SEC

AXR\$

\$(1)
HRGU*

01	000000	10	00	00	00	0	000006	LA	A0,I2
	000001	30	00	00	00	0	000002	MI	A0,K2
	000002	71	10	00	00	0	000003	DA	A0,K3-1
	000003	42	00	01	00	0	000000	AND	A1,MASK
	000004	30	00	02	00	0	000002	MI	A2,K2
	000005	71	10	02	00	0	000003	DA	A2,K3-1
	000006	42	00	03	00	0	000000	AND	A3,MASK
	000007	01	00	04	00	0	000006	SA	A4,I2
	000010	10	00	00	00	0	000005	LA	A0,I1
	000011	30	00	00	00	0	000001	MI	A0,K1
	000012	14	00	01	00	0	000007	AA	A1,(1)
	000013	42	00	01	00	0	000000	AND	A1,MASK
	000014	01	00	02	00	0	000005	SA	A2,I1
	000015	73	06	02	00	0	000016	LSC	A2,A2
	000016	73	12	02	00	0	000001	LSSL	A2,1
	000017	10	00	01	00	0	000010	LA	A1,(0200)
	000020	15	00	01	00	0	000017	ANA	A1,A3
	000021	73	03	01	00	0	000011	DSL	A1,9
	000022	73	06	04	00	0	000020	LSC	A4,A4
	000023	73	12	04	00	0	000001	LSSL	A4,1
	000024	10	00	03	00	0	000010	LA	A3,(0200)
	000025	15	00	03	00	0	000021	ANA	A3,A5
	000026	73	03	03	00	0	000011	DSL	A3,9
	000027	01	00	02	13	1	000000	SA	A2,*0,X11
	000030	01	00	04	13	1	000001	SA	A4,*1,X11
	000031	74	04	00	13	0	000003	J	3,X11

\$(0)

MASK

K1

K2

K3

I1

I2

00

000000

000001

000002

000003

000004

000005

000006

000007

000010

037777777777

4097

129

0

27075473833

20867350019

18575103187

END

FOR 6 01 JUN 72 10:55:51
UNIVAC 1108 FORTRAN V LEVEL 2206 0018 F5018P
THIS COMPILATION WAS DONE ON 01 JUN 72 AT 10:55:51

SUBROUTINE RG ENTRY POINT 000153

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000160
0000 *DATA 000027
0002 *BLANK 000000

EXTERNAL REFERENCES (BLOCK, NAME)

0003 RGU
0004 ALOG
0005 SQR
0006 COS
0007 NERR3\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

Block	Type	Relative Location	Name
0001	R	000024 10L	RGU
0004	R	000000 ALOG	ALOG
0000	I	000005 IB2	IB2
0001	R	000030 20L	0001
0000	R	000007 A1	0000 R 000007 A1
0000	I	000001 IFLAG	0000 I 000001 IFLAG
0001	R	000044 30L	0001
0000	R	000006 A1S	0000 R 000006 A1S
0000	R	000000 PI	0000 R 000000 PI
0001	R	000050 40L	0001
0000	R	000010 A2	0000 R 000010 A2
0000	R	000002 U1	0000 R 000002 U1
0001	R	000141 50L	0001
0000	I	000004 IR1	0000 I 000004 IR1
0000	R	000003 U2	0000 R 000003 U2

00101 1* SUBROUTINE RG(X)
00103 2* DATA PI/3.14159265358979/
00105 3* DATA IFLAG/0/
00107 4* IF (IFLAG.GT.0) GO TO 50
00111 5* CALL RGU(U1,U2)
00112 6* IF (U1.LT..99) GO TO 10
00114 7* U1=(U1-.99)*100.0
00115 8* IB1=1
00116 9* GO TO 20
00117 10* U1=U1*1.01010101
00120 11* IB1=0
00121 12* IF (U2.LT..99) GO TO 30
00123 13* U2=(U2-.99)*100.0
00124 14* IB2=1
00125 15* GO TO 40
00126 16* U2=U2*1.01010101
00127 17* IB2=0
00130 18* A1S=-2.0*ALOG(U1)
00131 19* A1=SQR(A1S)
00132 20* A2=2.0*PI*U2
00133 21* U1=A1*COS(A2)
00134 22* U2=SQR(A1S-U1**2)
00135 23* IF (A2.GT.PI) U2=-U2
00137 24* U1=U1*6

00140	25*	U2=U2*6
00141	26*	IF (IB1.GT.0) U1=U1+7
00143	27*	IF (IB2.GT.0) U2=U2+7
00145	28*	X=U1
00146	29*	IFLAG=1
00147	30*	RETURN
00150	31*	50 X=U2
00151	32*	IFLAG=0
00152	33*	RETURN
00153	34*	END

0 *DIAGNOSTIC* MESSAGE(S)

END OF UNIVAC 1108 FORTRAN V COMPILATION.

PHASE 1	TIME =	0 SEC.
PHASE 2	TIME =	0 SEC.
PHASE 3	TIME =	1 SEC.
PHASE 4	TIME =	0 SEC.
PHASE 5	TIME =	0 SEC.
PHASE 6	TIME =	0 SEC.

TOTAL COMPILATION TIME = 1 SEC

AXR\$

\$(1)

RGU*

000001	000000	10	00	00	00	0	000006
000002	000001	30	00	00	00	0	000002
000003	000002	71	10	00	00	0	000003
000004	000003	42	00	01	00	0	000000
000005	000004	30	00	02	00	0	000002
000006	000005	71	10	02	00	0	000003
000007	000006	42	00	03	00	0	000000
000008	000007	01	00	04	00	0	000006
000009	000010	10	00	00	00	0	000005
000010	000011	30	00	00	00	0	000001
000011	000012	14	00	01	00	0	000007
000012	000013	42	00	01	00	0	000000
000013	000014	01	00	02	00	0	000005
000014	000015	73	06	02	00	0	000016
000015	000016	73	12	02	00	0	000001
000016	000017	10	00	01	00	0	000010
000017	000018	15	00	01	00	0	000017
000018	000019	73	03	01	00	0	000011
000019	000020	73	06	04	00	0	000020
000020	000021	73	12	04	00	0	000001
000021	000022	10	00	03	00	0	000010
000022	000023	15	00	03	00	0	000021
000023	000024	73	03	03	00	0	000011
000024	000025	01	00	02	13	1	000000
000025	000026	01	00	04	13	1	000001
000026	000027	74	04	00	13	0	000003

\$(0)

MASK

K1

K2

K3

I1

I2

00	000000	377777777777
000030	000001	00000010001
000031	000002	00000000201
000032	000003	00000000000
000033	000004	311564570651
000034	000005	233362477003
000035	000006	212312312323
000036	000007	0000000000001
000037	000010	0000000000200

LA	A0,I2
MI	A0,K2
DA	A0,K3-1
AND	A1,MASK
MI	A2,K2
DA	A2,K3-1
AND	A3,MASK
SA	A4,I2
LA	A0,I1
MI	A0,K1
AA	A1,(1)
AND	A1,MASK
SA	A2,I1
LSC	A2,A2
LSSL	A2,1
LA	A1,(0200)
ANA	A1,A3
DSL	A1,9
LSC	A4,A4
LSSL	A4,1
LA	A3,(0200)
ANA	A3,A5
DSL	A3,9
SA	A2,*0,X11
SA	A4,*1,X11
J	3,X11
+	03777777777777
+	4097
+	129
+	0
+	27075473833
+	20867350019
+	18575103187
+	END

SUBROUTINE GRAPH ENTRY POINT 000536

STORAGE USED (BLOCK, NAME, LENGTH)

0001 *CODE 000571
0000 *DATA 000572
0002 *BLANK 000000

EXTERNAL REFERENCES (BLOCK, NAME)

0003 NWDJ\$
0004 NI01\$
0005 NI02\$
0006 NERR2\$
0007 NERR3\$

STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

0000	000455	100F	0001	000151	109L	0001	000154	110L	0001	000050	125G	0001	000064	134G
0001	000130	152G	0001	000205	207G	0001	000215	213G	0001	000220	216G	0001	000232	225G
0001	000261	240G	0001	000304	240L	0001	000312	250L	0001	000320	260L	0001	000346	272L
0001	000334	274G	0001	000351	275L	0001	000353	278L	0001	000363	280L	0001	000500	290F
0001	000407	300L	0000	000504	310F	0001	000402	322G	0001	000415	332G	0001	000434	345G
0001	000421	350L	0001	000456	357G	0001	000470	365G	0001	000425	400L	0000	000510	420F
0001	000014	910L	0000	000416	915F	0000	000453	916F	0000	000425	917F	0000	000434	918F
0000	I 000363	BLANK	0000	I 000364	I	0000	I 000405	IFLAG	0000	I 000365	IO	0000	I 000401	IOUT
0000	I 000410	ITA	0000	I 000216	ITOTAL	0000	I 000414	I\$	0000	I 000371	I4	0000	I 000403	J
0000	I 000404	K	0000	I 000407	KA	0000	I 000412	KK	0000	I 000402	L	0000	I 000411	LA
0000	I 000370	LJMA	0000	I 000366	LJML	0000	I 000367	LIMU	0000	I 000205	NUMB	0000	I 000000	PRINT
0000	I 000153	SYM	0000	R 000373	XDELTA	0000	R 000400	XHIGH	0000	R 000377	XL	0000	R 000145	XP
0000	R 000415	XR	0000	R 000406	XT	0000	R 000372	YDELTA	0000	R 000374	YL	0000	R 000376	YLOW
0000	R 000413	YP	0000	R 000375	YT									

00101	1*	SUBROUTINE GRAPH(N,X,Y,IWRD,XMAX,XMIN,YMAX,YMIN)
00103	2*	DIMENSION X(1),Y(1),PRINT(101),XP(6),SYM(26)
00104	3*	DIMENSION NUMB(6),ITOTAL(101)
00105	4*	INTEGER PRINT,BLANK,SYM
00106	5*	DATA BLANK/1H /
00110	6*	DATA (NUMB(1),I=1,9)/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
00112	7*	DATA (SYM(1),I=1,26)/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,
00112	8*	* 1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
00112	9*	C IO = NUMBER OF PRINT TAPE
00114	10*	IO = 6
00115	11*	LJML=1
00116	12*	LIMU=26
00117	13*	910 IF (LIMU.GT.N) LIMU=N


```

000000 15*
00122 16*
00131 17*
00132 18*
00140 19*
00141 20*
00142 21*
00143 22*
00144 23*
00145 24*
00146 25*
00147 26*
00150 27*
00151 28*
00154 29*
00157 30*
00162 31*
00165 32*
00170 33*
00171 34*
00173 35*
00177 36*
00177 37*
00200 38*
00202 39*
00203 40*
00205 41*
00205 42*
00206 43*
00211 44*
00212 45*
00215 46*
00220 47*
00221 48*
00223 49*
00224 50*
00227 51*
00232 52*
00235 53*
00236 54*
00237 55*
00242 56*
00245 57*
00250 58*
00253 59*
00254 60*
00255 61*
00256 62*
00257 63*
00260 64*
00261 65*
00262 66*
00264 67*
00266 68*
00267 69*
00270 70*
00273 71*
00276

LIMU-+1
WRITE(6,915) IWRD,(I4,I4=LIML,LIMU)
915 FORMAT('I4',I4,' POSITIONS OF UNITS',26I4)
WRITE(6,917) (SYM(I4),I4=1,LIMA)
917 FORMAT(' ASSOCIATED SYMBOLS ARE',26(3X,A1))
YDELTA=(YMAX-YMIN)/50.
XDELTA=(XMAX-XMIN)/100.
YL =YMAX-YDELTA/2.
YT=YMAX+YDELTA/2.
YLOW=YMIN-YDELTA/2.
XL=XMIN-XDELTA/2.
XHIGH=XMAX+XDELTA/2.
IOUT=0
DO 110 I=LIML,LIMU
IF (Y(I)-YT) 101,101,109
101 IF (Y(I)-YLOW) 109,109,102
102 IF (X(I)-XL) 109,103,103
103 IF (X(I)-XHIGH) 110,109,109
109 IOUT=IOUT+1
110 CONTINUE
918 IF (IOUT.GT.0) WRITE(6,918) IOUT
918 FORMAT(' * * * NOTE,'I4',I4, ' POINTS ARE OUTSIDE THE LIMITS OF THE GRA
*PH AND HAVE BEEN OMITTED')
WRITE(6,916)
916 FORMAT(' ', ' ')
WRITE(10,100)
100 FORMAT(14X,101H+-----+-----+-----+-----+
1-----+-----+-----+-----+-----+
DO 350 I=1,6
L=1
DO 350 J=1,10
DO 200 K=1,101
ITOTAL(K)=1
200 PRINT(K)=BLANK
IFLAG=0
DO 260 K=LIML,LIMU
IF (Y(K)-YT) 205,205,260
205 IF (Y(K)-YL) 260,260,210
210 XL=XMIN
XT=XMIN+XDELTA/2.
DO 255 KA=1,101
IF (X(K)-XL) 250,215,215
215 IF (X(K)-XT) 220,250,250
220 IF (PRINT(KA)-BLANK) 240,230,240
230 ITA=K-(K-1)/26*26
PRINT(KA)=SYM(ITA)
GO TO 260
240 ITOTAL(KA)=ITOTAL(KA)+1
IFLAG=1
GO TO 260
250 XL=XT
255 XT=XT+XDELTA
260 CONTINUE
YT=YL
YL=YL-YDELTA
IF (IFLAG) 265,278,265
265 DO 275 LA=1,101
IF (ITOTAL(LA)-1) 268,275,268

```

100
110
120

140
150
160
170
175

210
220

250
260

00301 72*
 00302 73*
 00305 74*
 00306 75*
 00307 76*
 00311 77*
 00312 78*
 00315 79*
 00316 80*
 00317 81*
 00326 82*
 00327 83*
 00330 84*
 00336 85*
 00337 86*
 00342 87*
 00351 88*
 00353 89*
 00354 90*
 00355 91*
 00356 92*
 00361 93*
 00363 94*
 00371 95*
 00372 96*
 00374 97*
 00375 98*
 00376 99*
 00377 100*

269 KK=ITOTAL(LA)
 IF (KK-9) 272,272,270
 270 KK=9
 272 PRINT(LA)=NUMB(KK)
 275 CONTINUE
 279 GO TO (280,300),L
 280 IF(I-5) 285,285,400
 285 L=2
 YP=YT+YDELTA/2.
 WRITE (IO,290) YP,PRINT
 290 FORMAT(1X,E12.4,1H+,101A1,1H+)
 GO TO 350
 300 WRITE (IO,310) PRINT
 310 FORMAT (13X,1H-,101A1,1H-)
 350 CONTINUE
 400 WRITE (IO,290) YMIN,PRINT
 WRITE (IO,100)
 XP(1)=XMIN
 XP(6)=XMAX
 XR=20.*XDELTA
 DO 410 I=2,5
 410 XP(I)=XP(I-1)+XR
 WRITE (IO,420) XP
 420 FORMAT(6(7X,E13.5))
 IF (LIMU.EQ.N) RETURN
 LIML=LIML+26
 LIMU=LIMU+26
 GO TO 910
 END

350
 360
 370
 380
 390
 400
 410
 420
 430
 440
 450
 460
 480
 490
 500
 510
 530

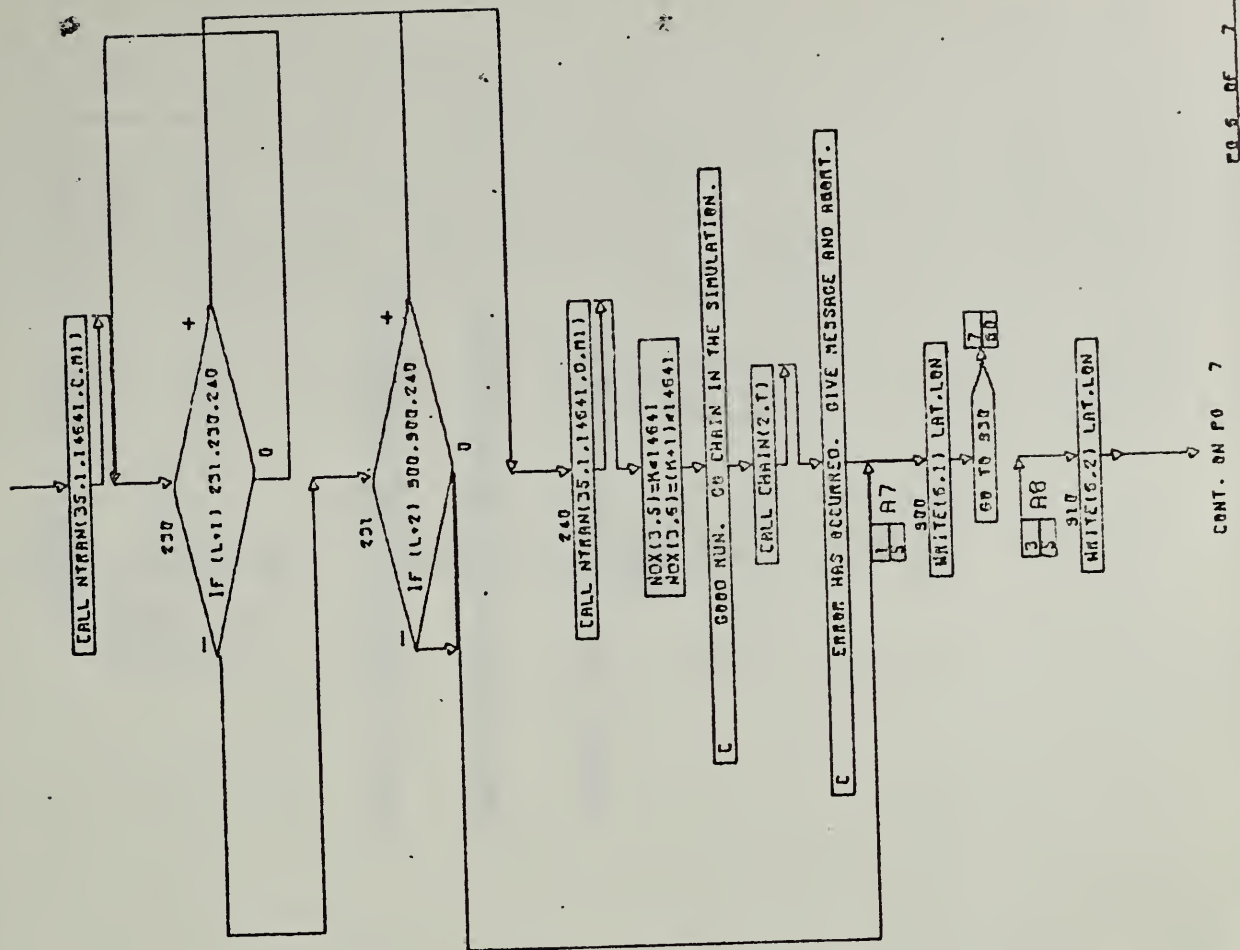
0 *DIAGNOSTIC* MESSAGE(S)

END OF UNIVAC 1108 FORTRAN V COMPILATION.

PHASE 1 TIME = 0 SEC.
 PHASE 2 TIME = 0 SEC.
 PHASE 3 TIME = 0 SEC.
 PHASE 4 TIME = 0 SEC.
 PHASE 5 TIME = 1 SEC.
 PHASE 6 TIME = 0 SEC.

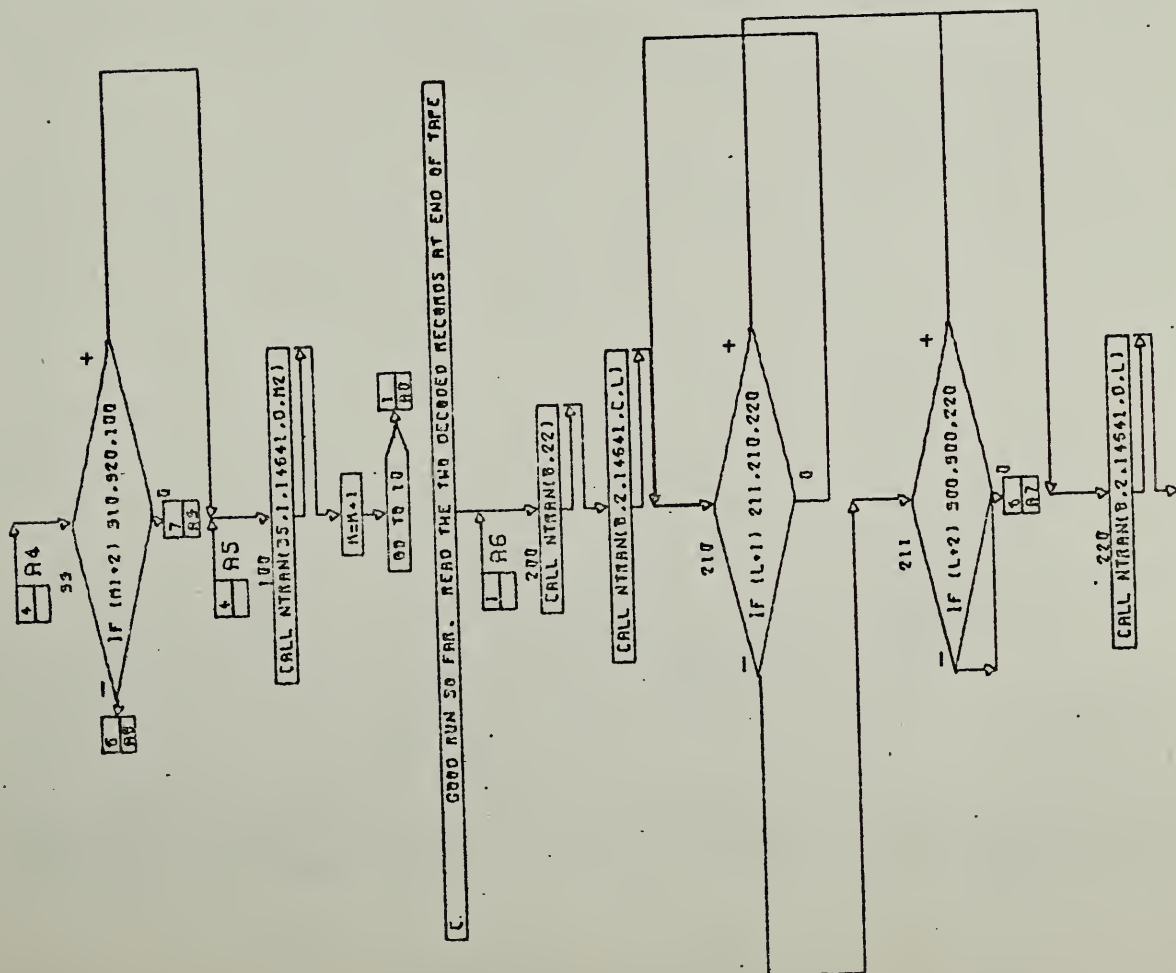
TOTAL COMPILATION TIME = 1 SEC

APPENDIX C: FLOWCHARTS OF THE MAIN SIMULATION PROGRAMS



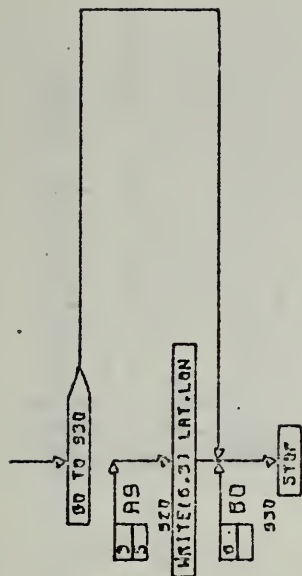
CONT. ON P0 7

P0 5 OF 7



CONT. ON P0 6

P0 5 OF 7

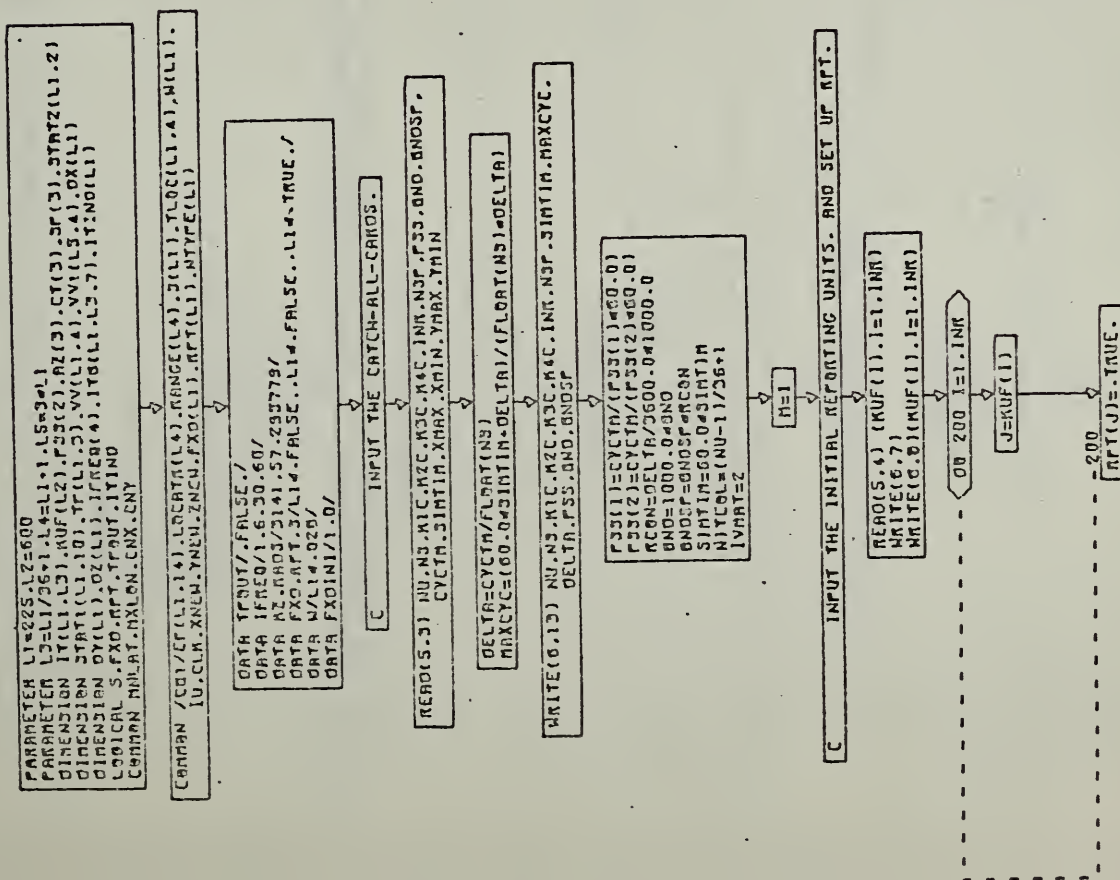
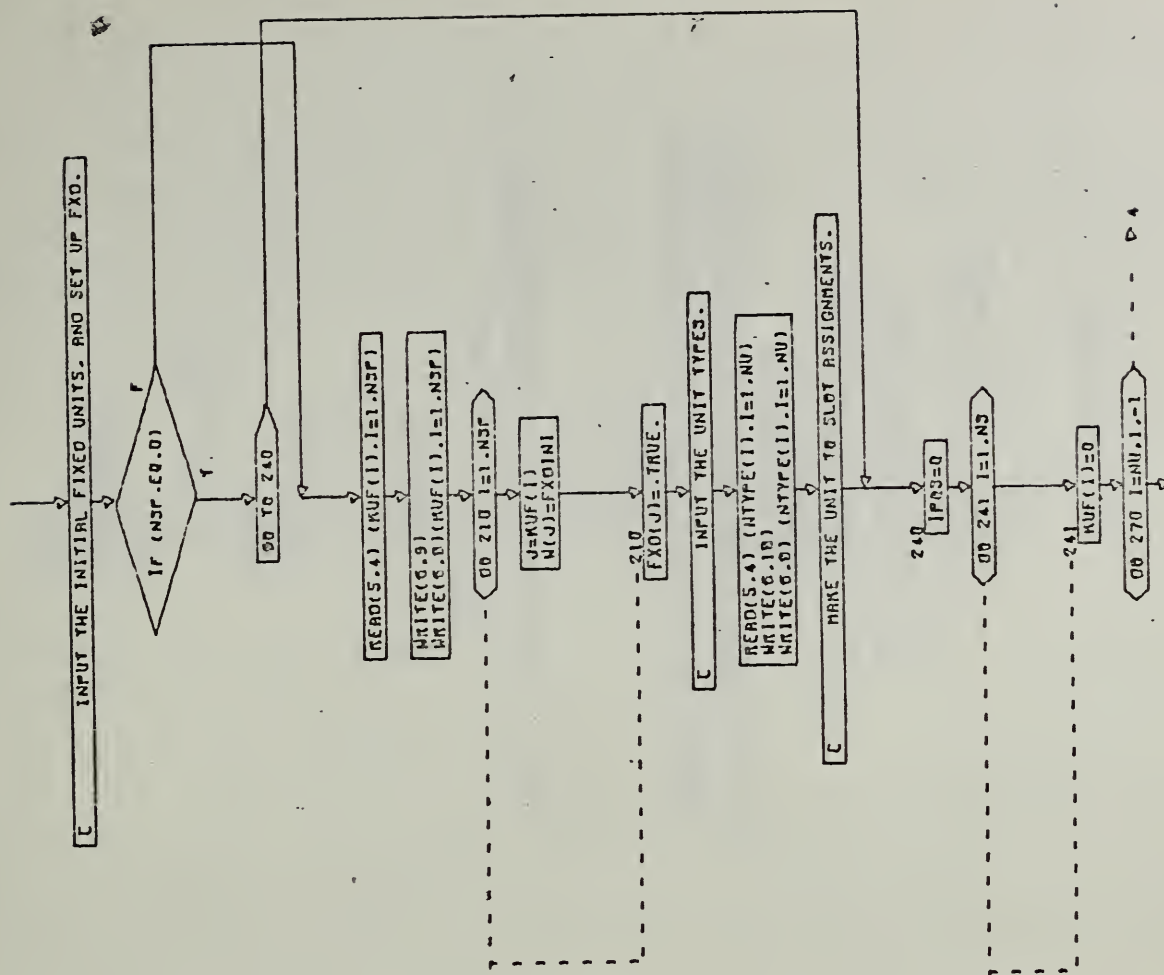


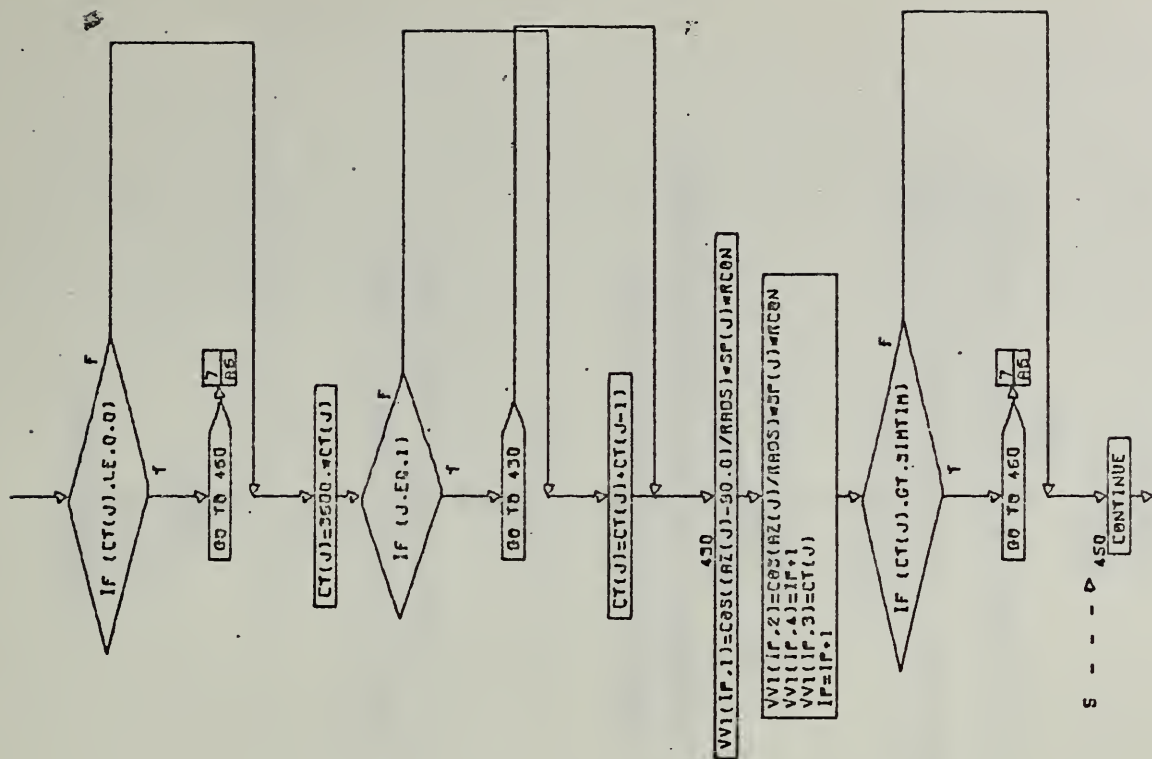
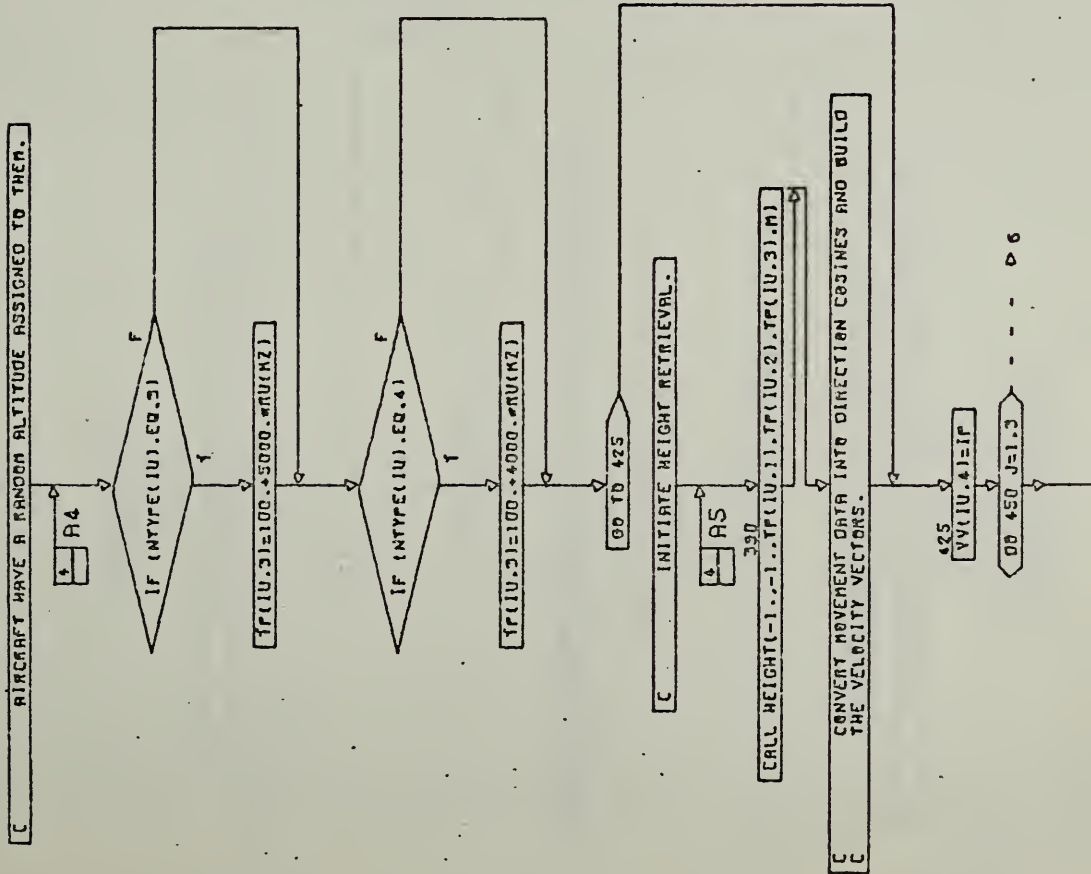
1
FORMAT(/' BPO TAPE READ. LAST READ WAS '2110)

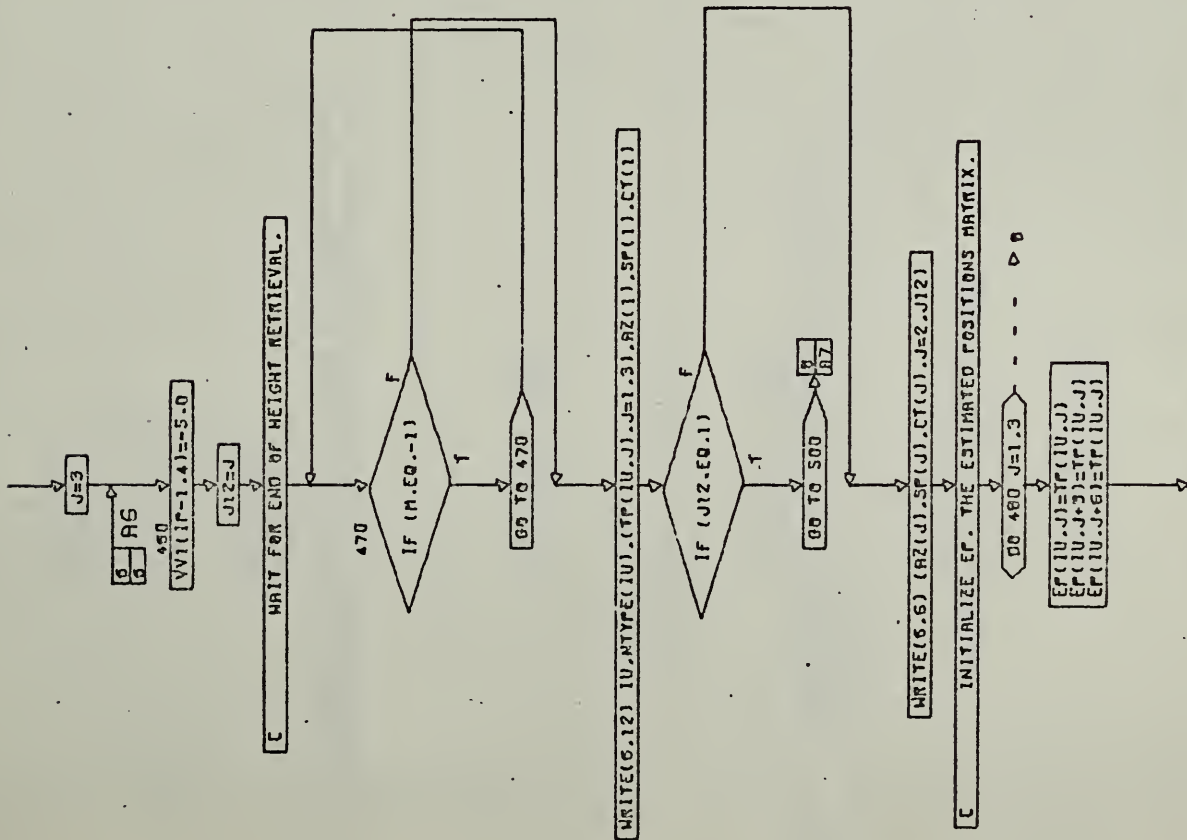
2
FORMAT(/' BPO OKUM WRITE. CURRENT IS '2110)

3
FORMAT(/' EOF ON OKUM. CURRENT IS '2110)

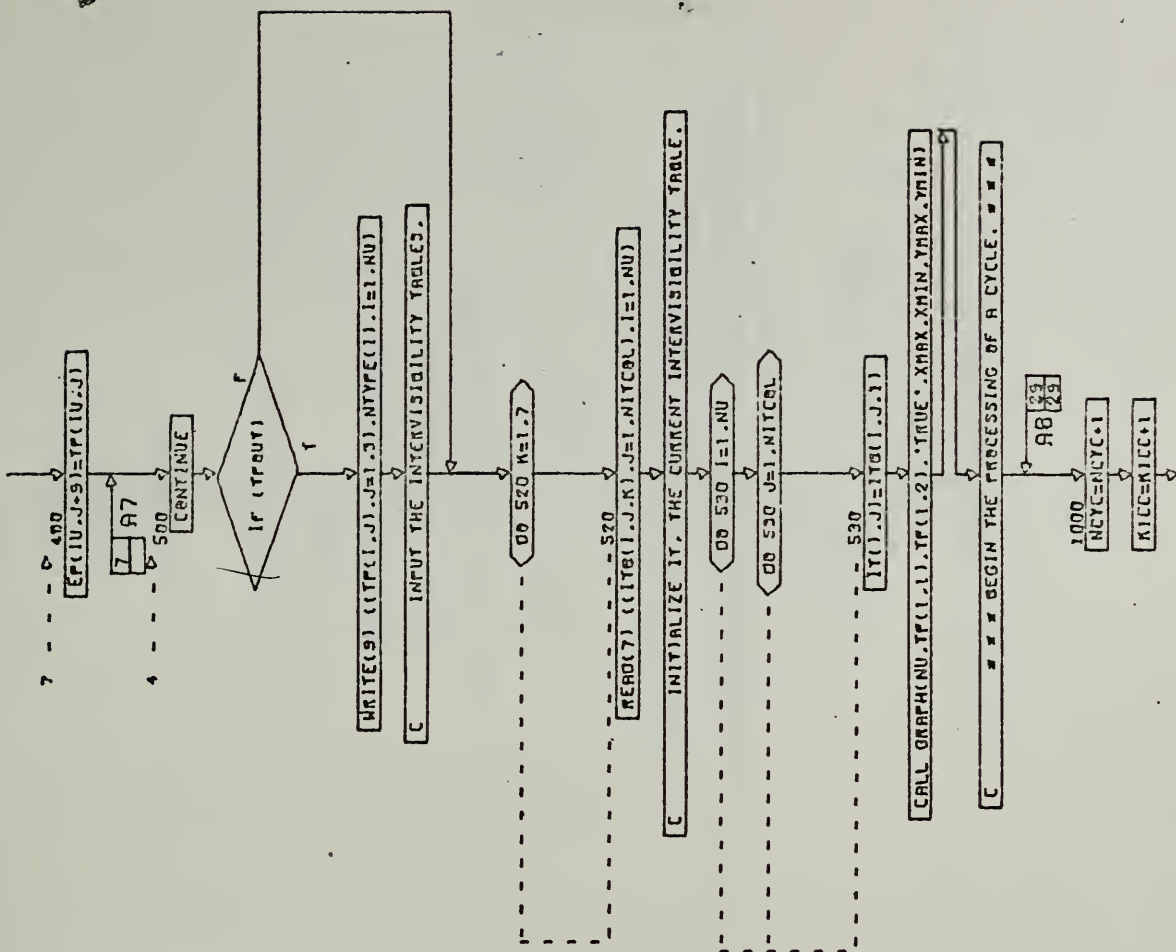
END



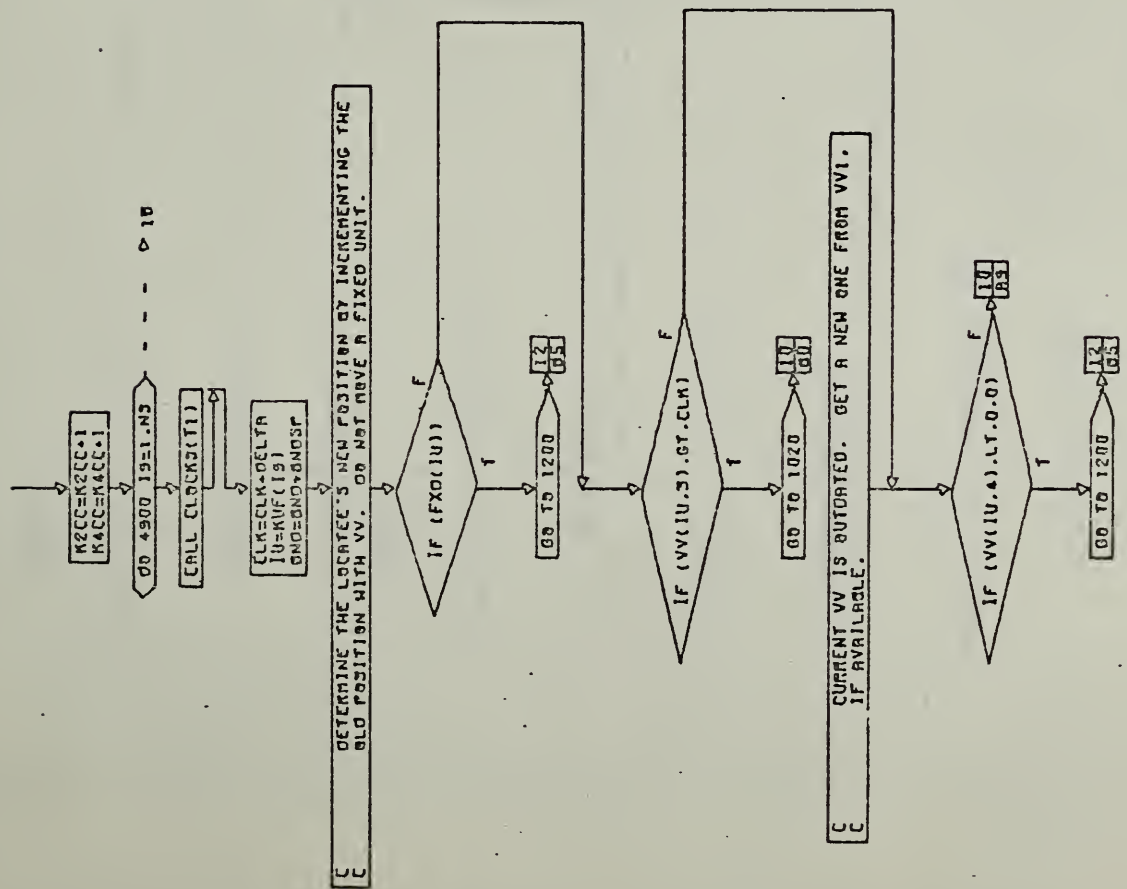
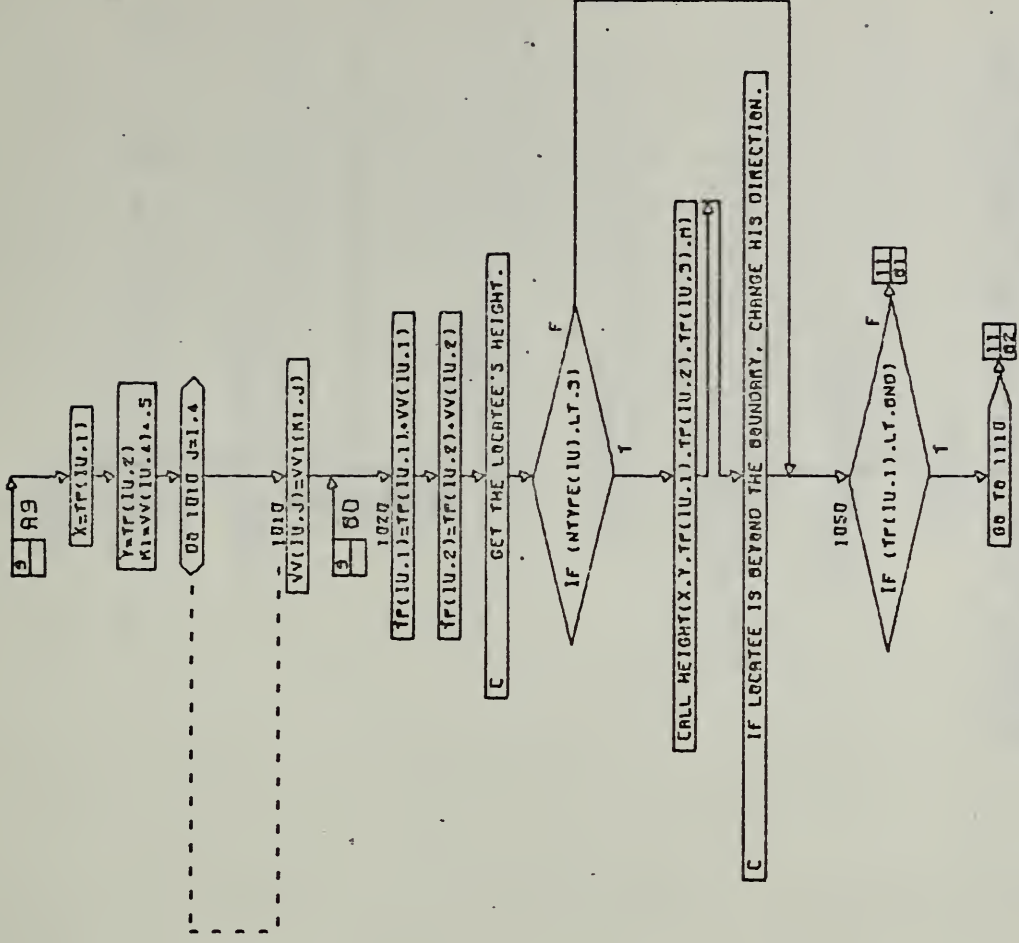


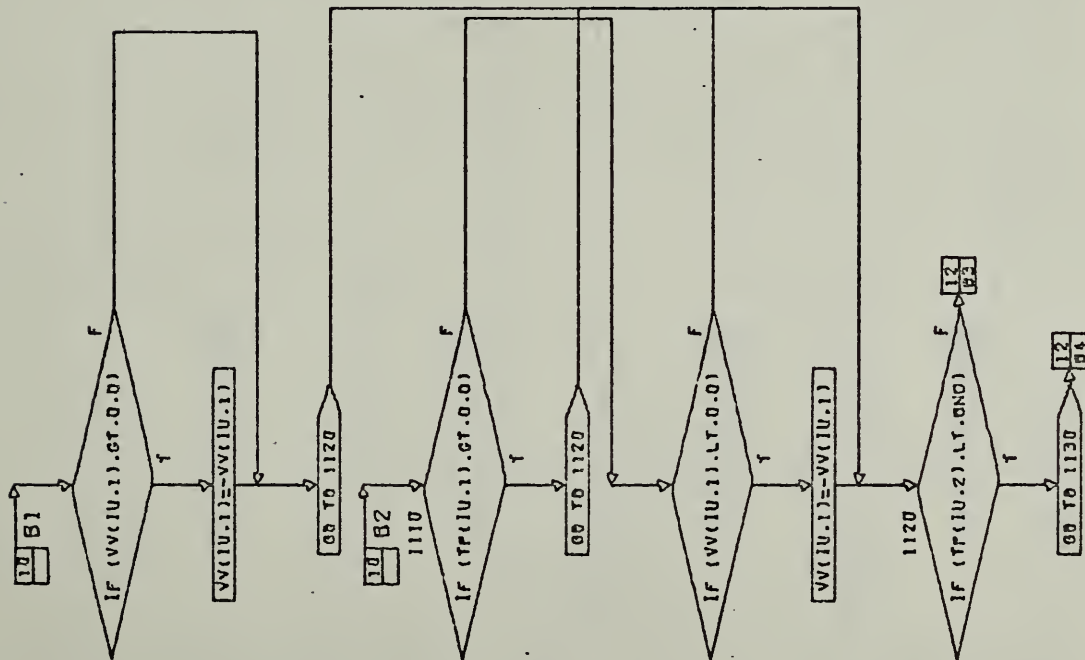


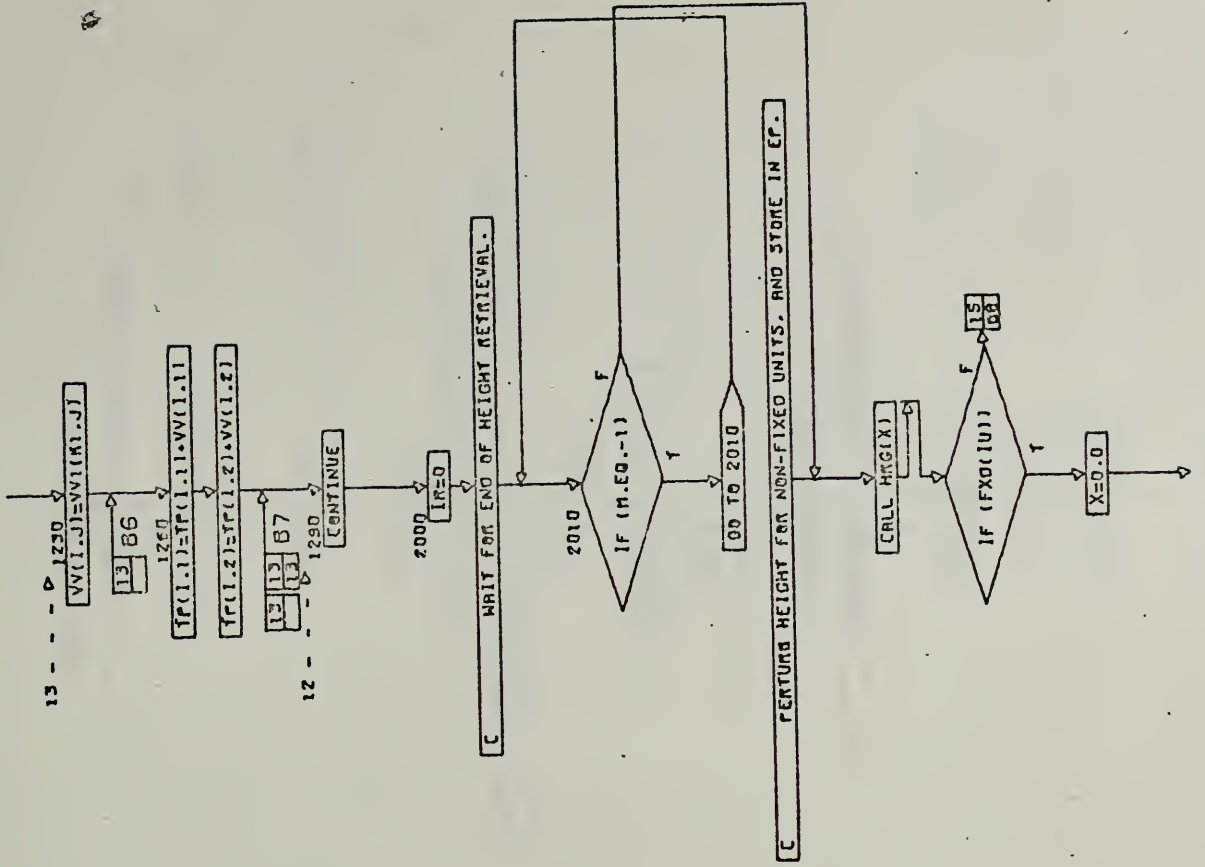
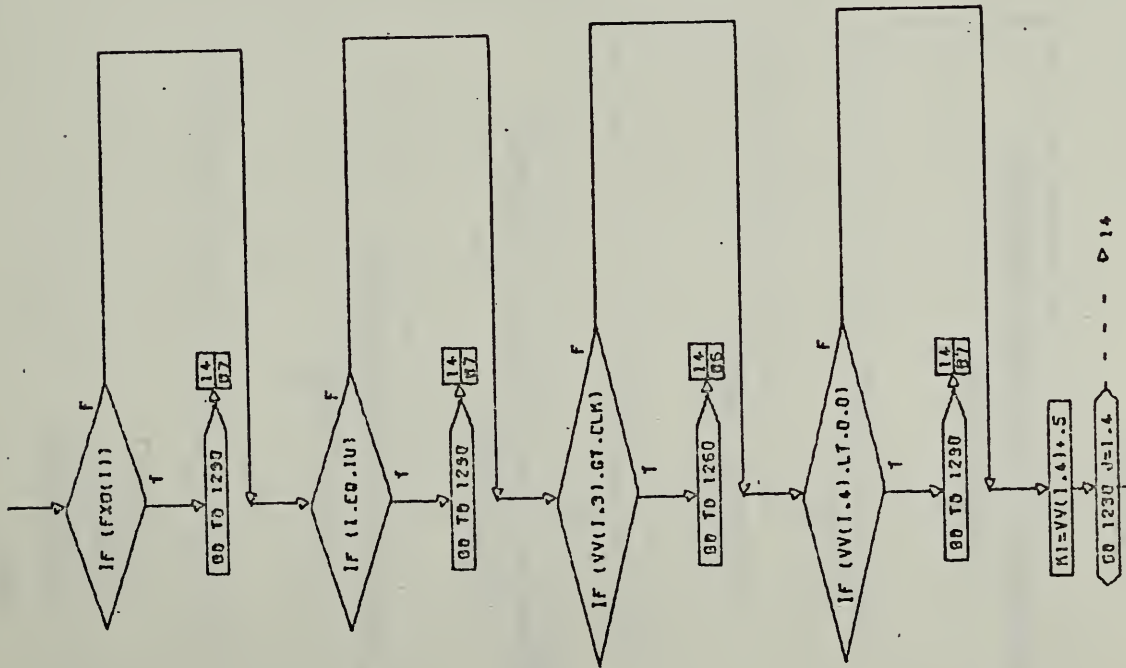
CONT. ON PG 8

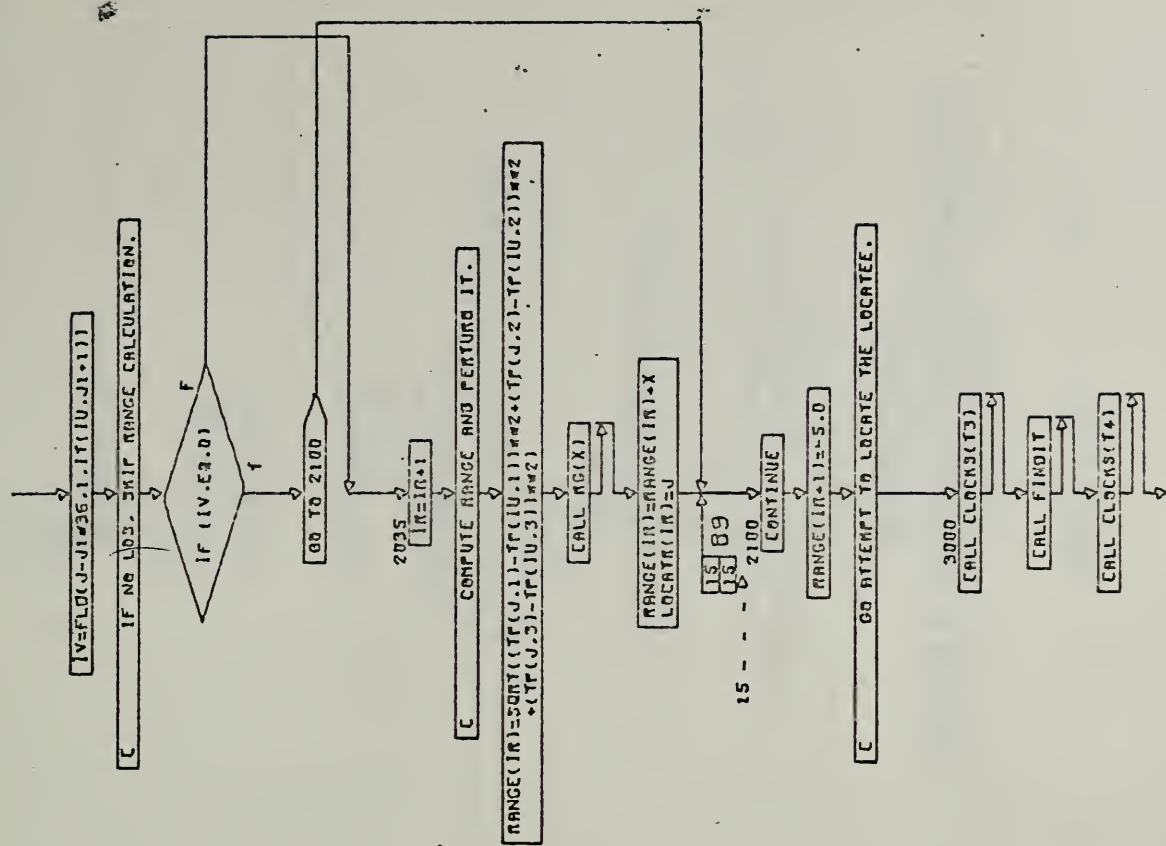
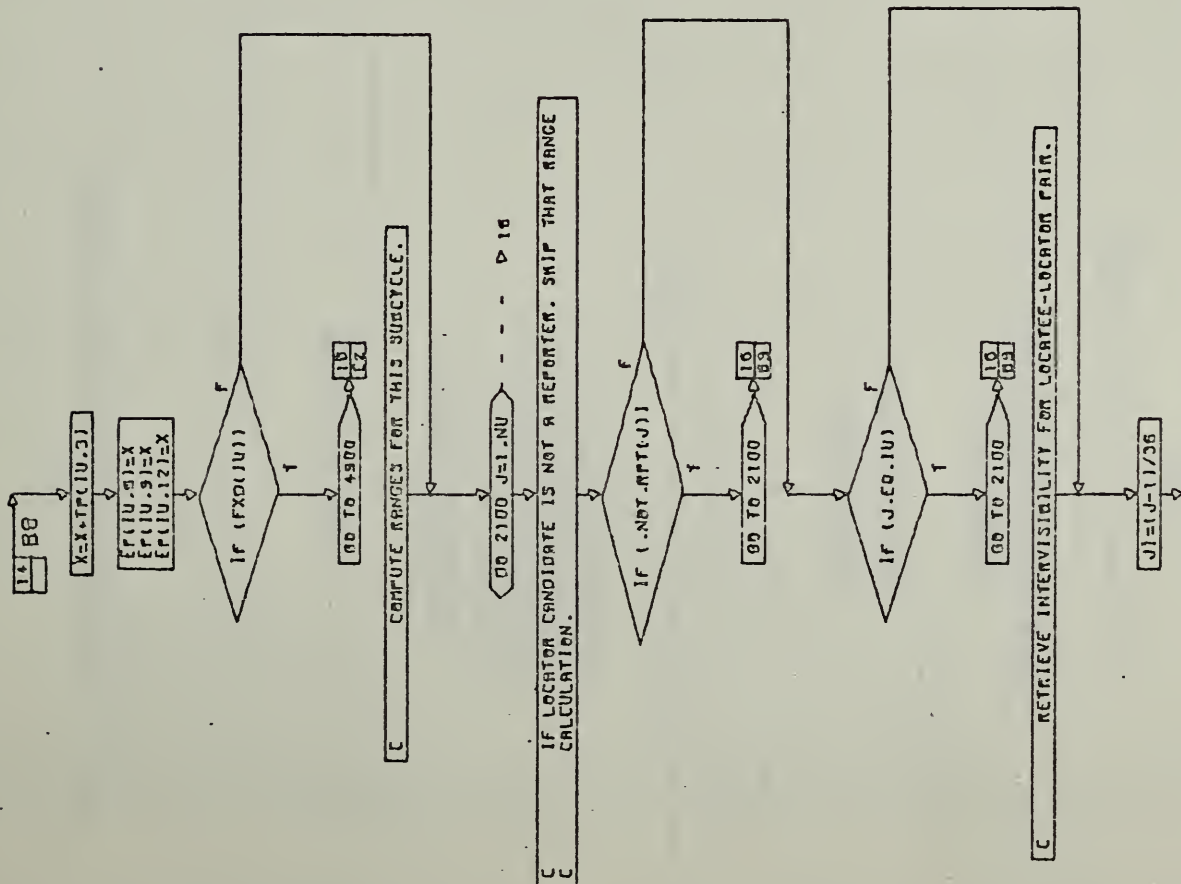


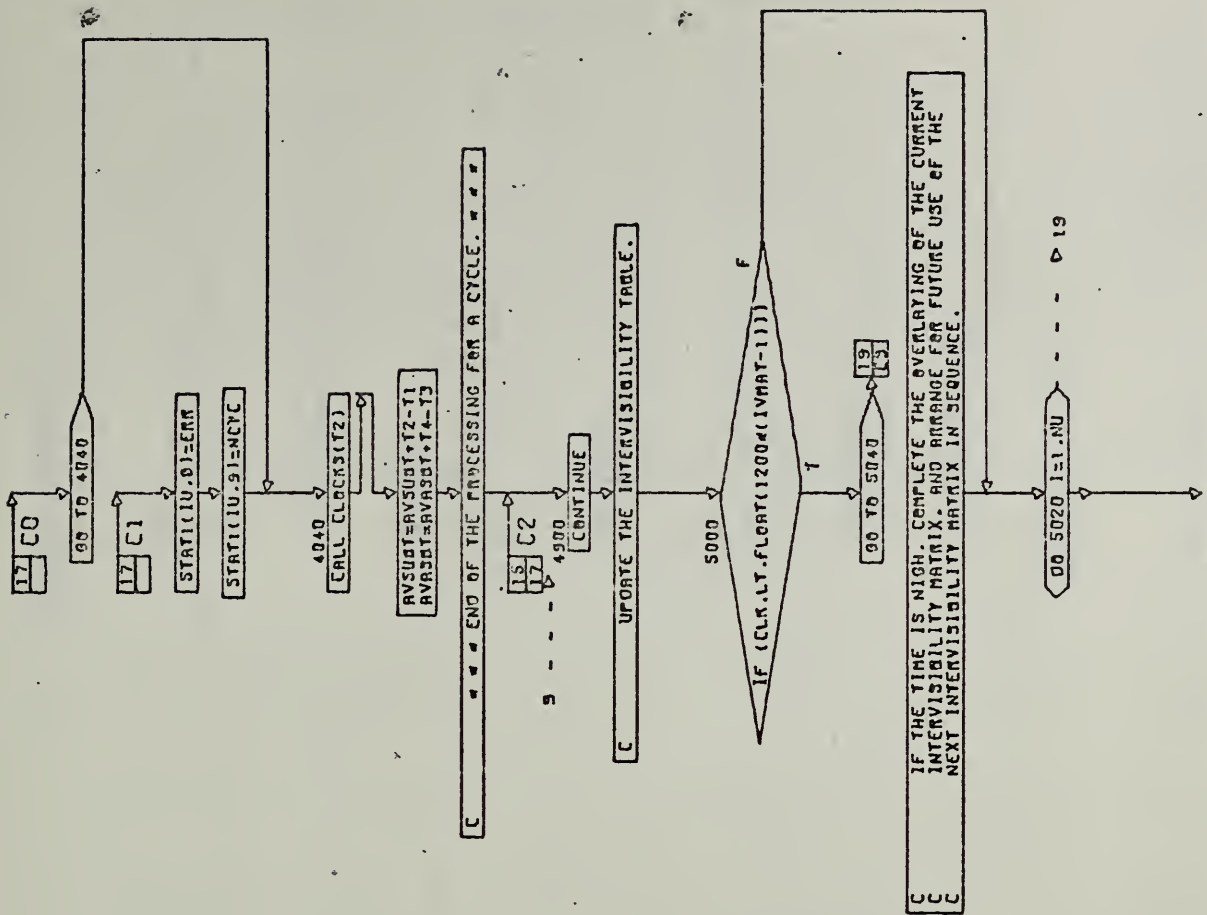
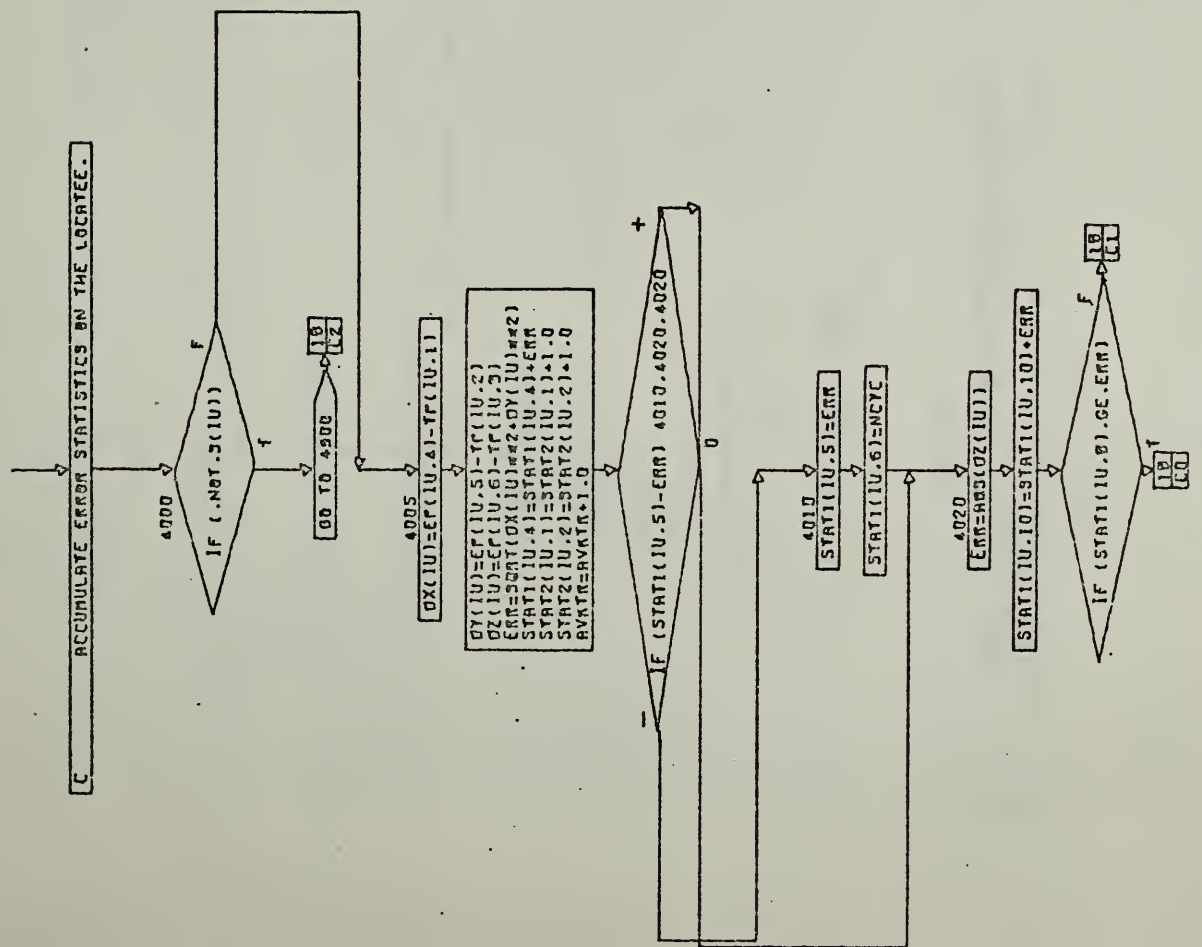
CONT. ON PG 9

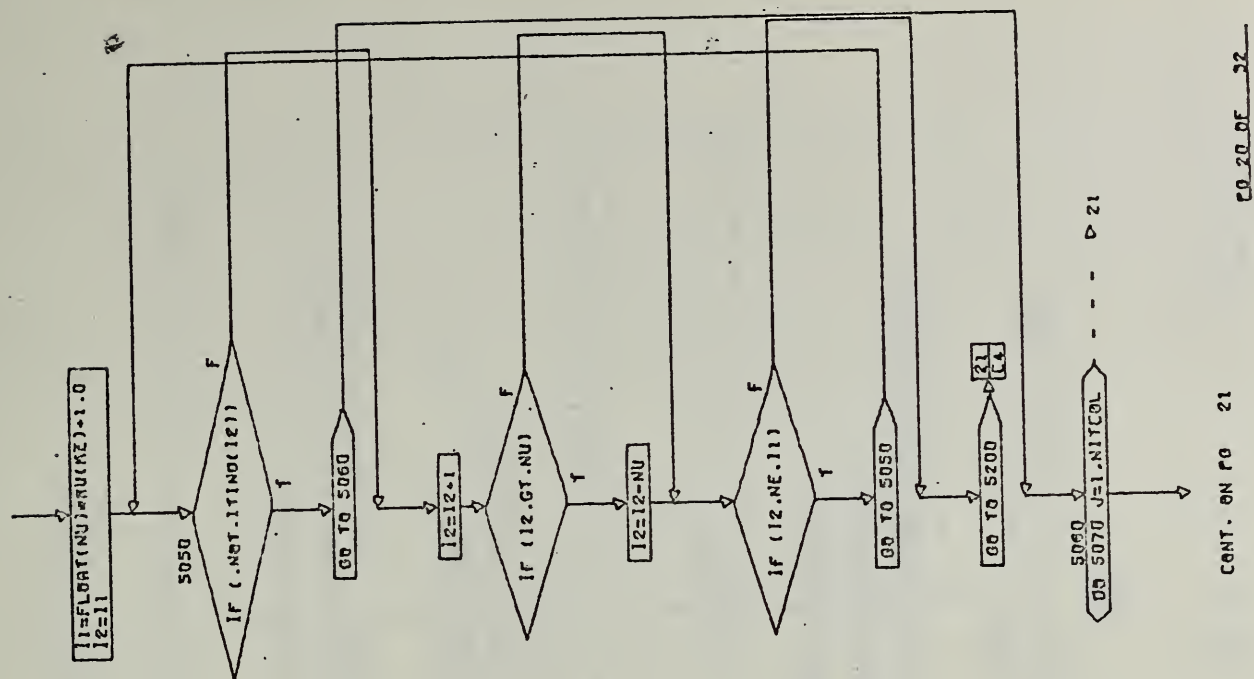
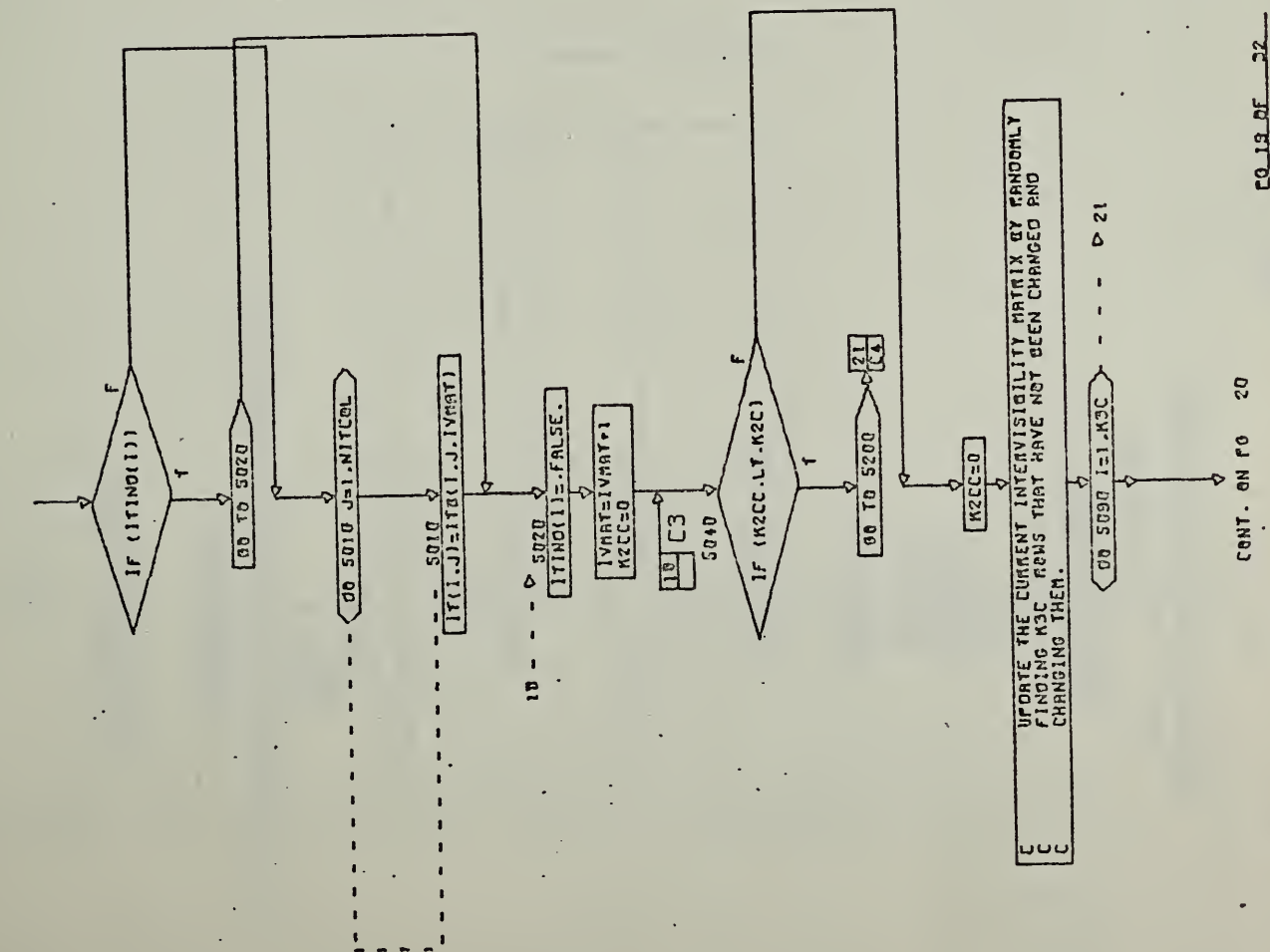


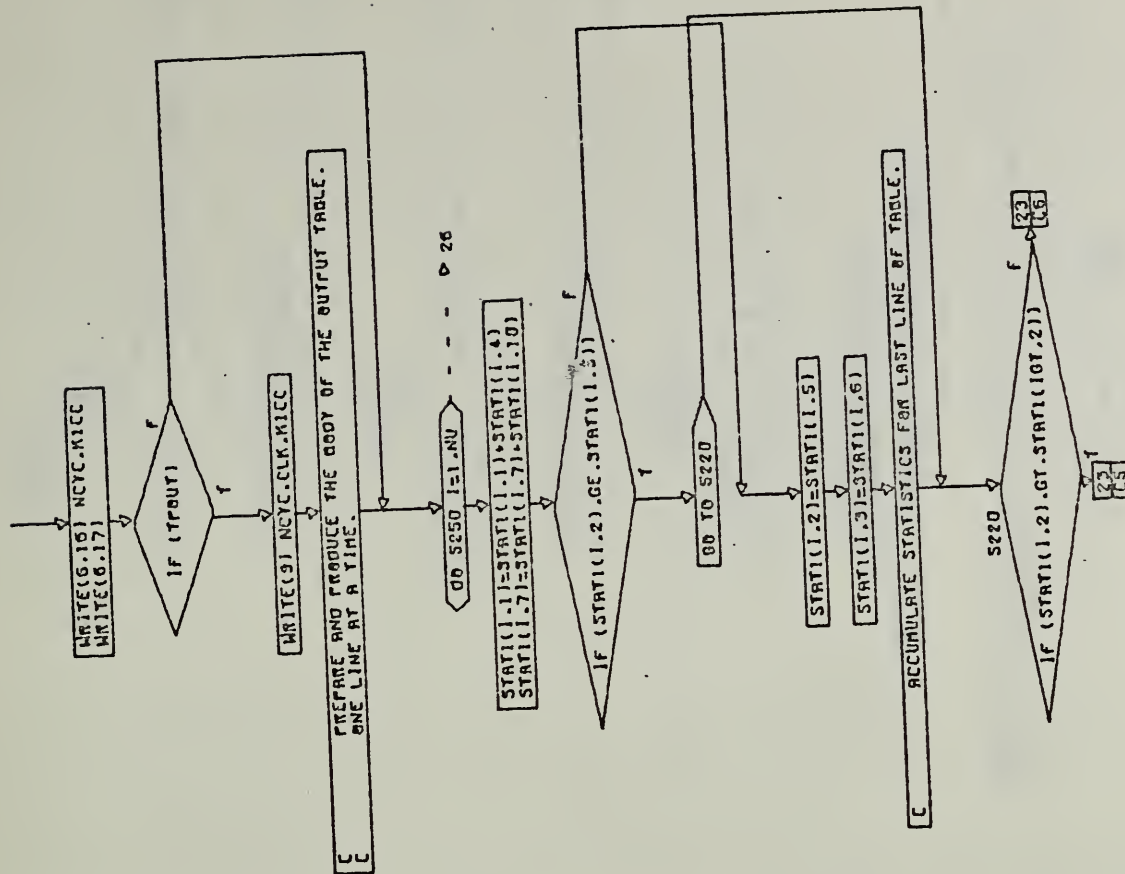






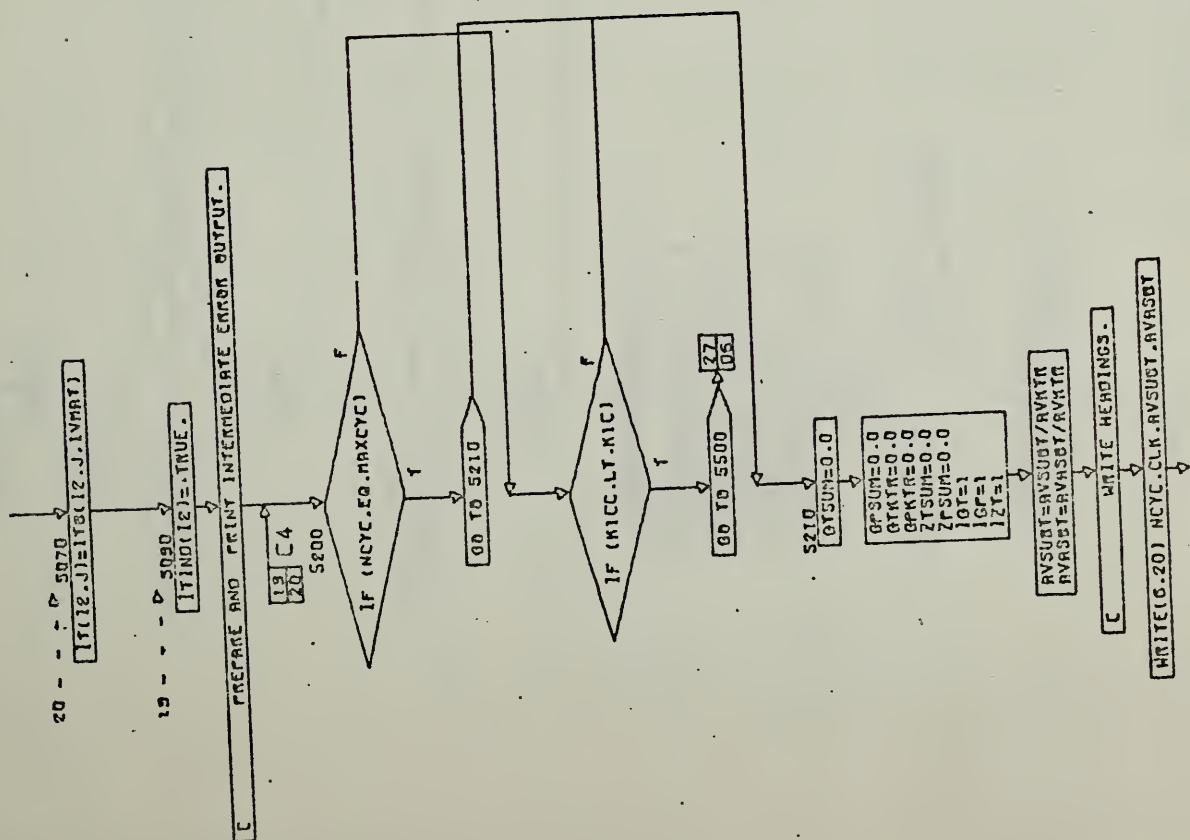






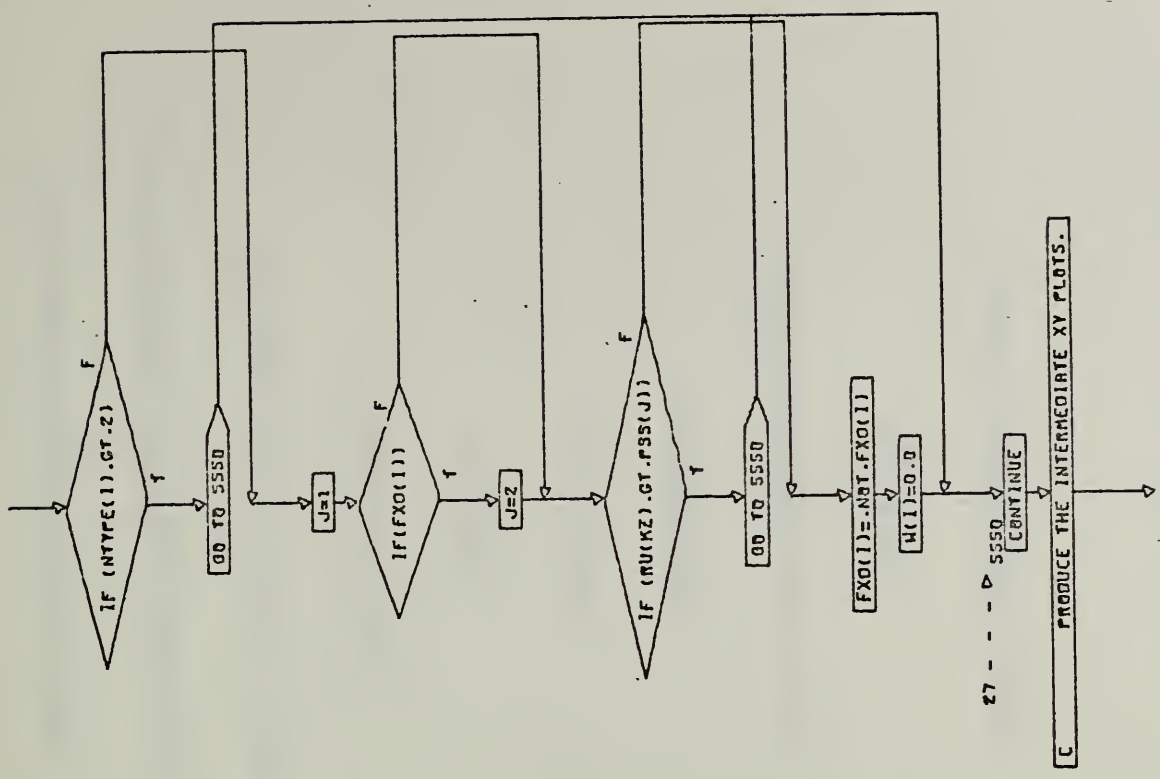
CONT. ON P0 23

P0.22 OF 32



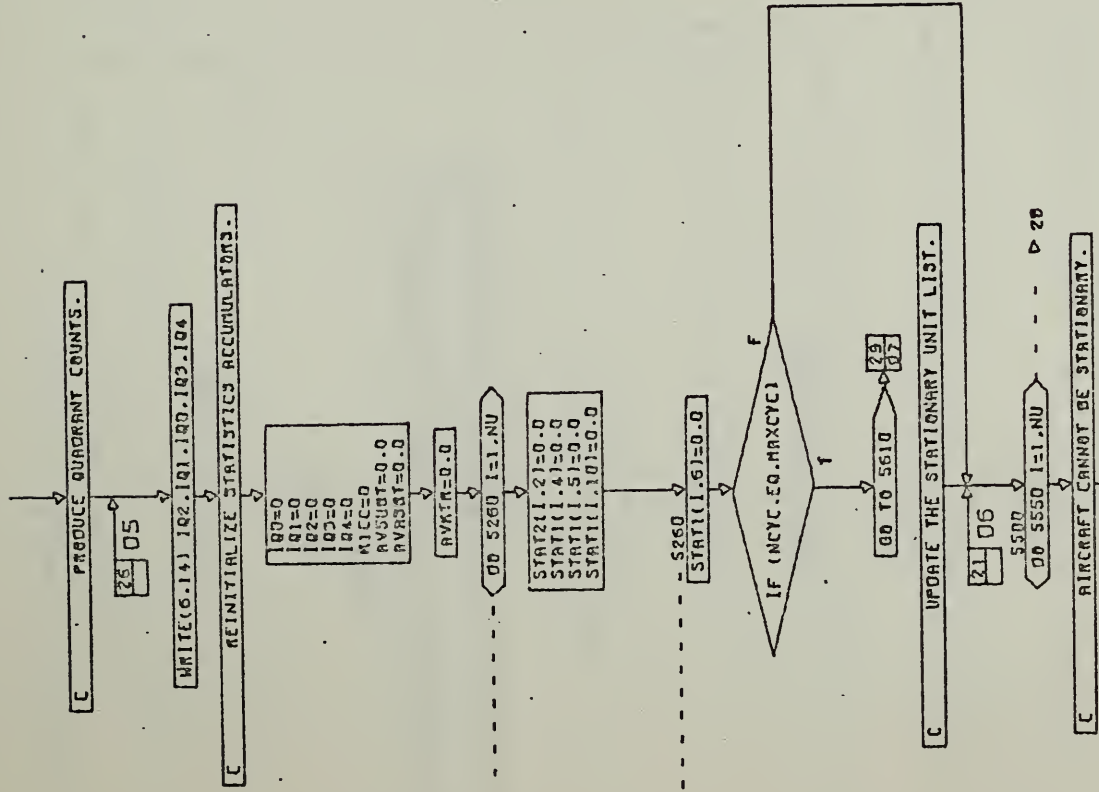
CONT. ON P0 22

P0.21 OF 32



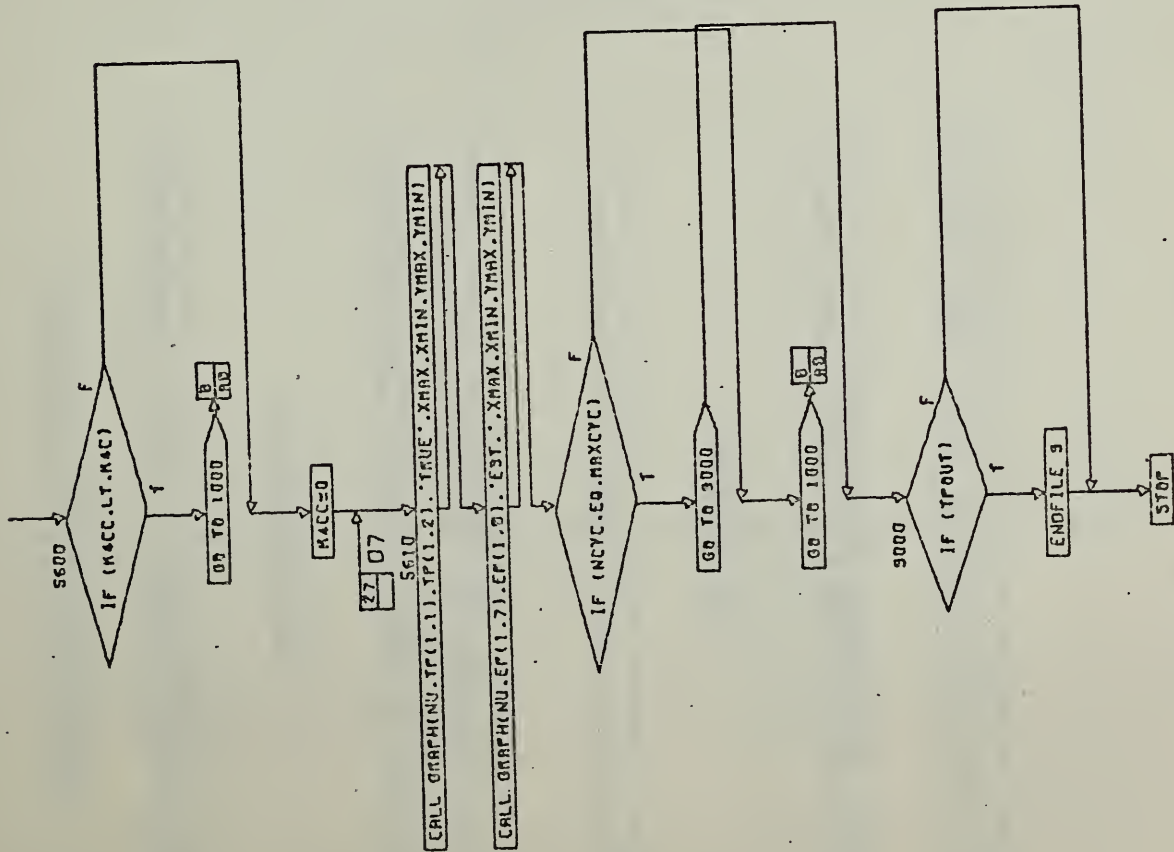
CONT. ON P0 29

CO 20 OF 32

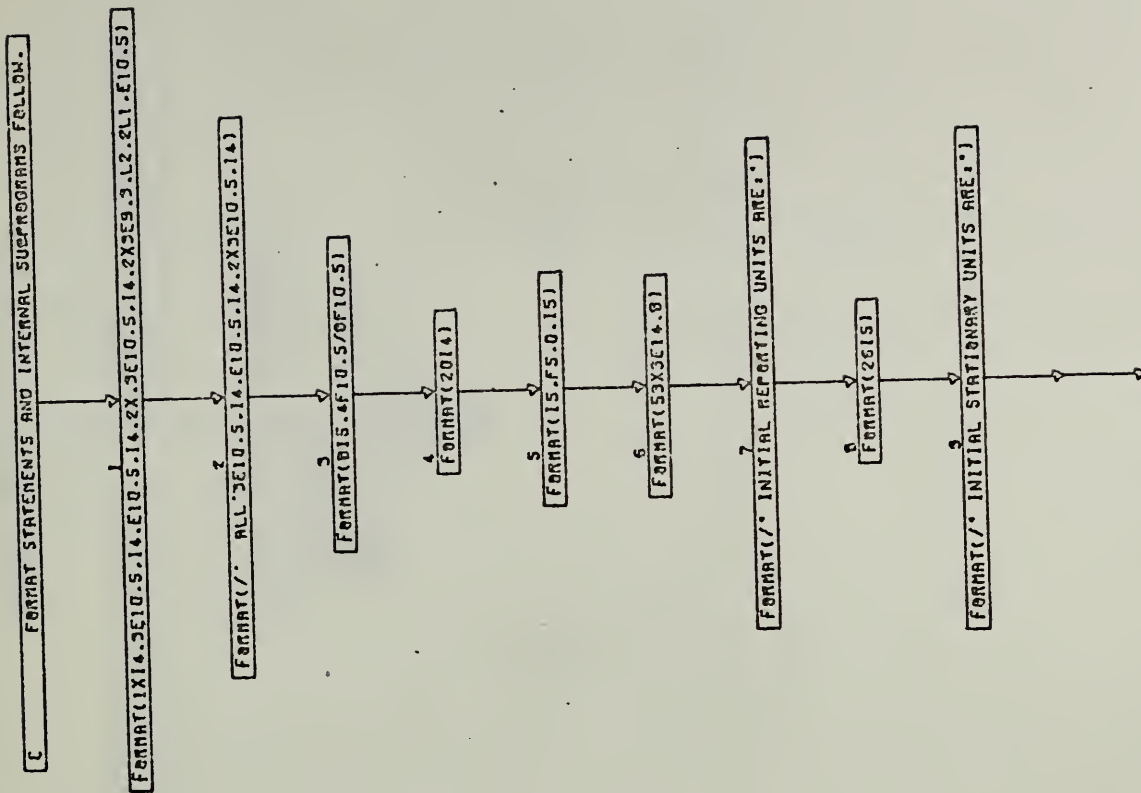


CONT. ON P0 20

CO 27 OF 32



CONT. ON P0 30





SUBROUTINE HEIGHT(PX, PY, FX, FY, FZ, M)
 PARAMETER LG=5, L7=11
 COMMON /MLAT, MLON, CNX, CNY, NOX(LG, L7)
 DATA CNX, CNY, /0145015310, .0197758475/

CONVERT FROM POSITION IN METERS TO POSITION IN TERMS OF
 THREE SECONDS OF LATITUDE AND LONGITUDE.

IFX=FX/CNX*.5
 IFY=FY/CNY*.5
 IFX=FX/CNX*.5
 IFY=FY/CNY*.5

IF (IFX.NE.IFX)

GO TO 100

IF (IFY.NE.IFY)

GO TO 100

IF FUTURE POSITION IS ON SAME TERRAIN PLATFORM AS PRESENT
 POSITION. FEED BACK SAME HEIGHT.

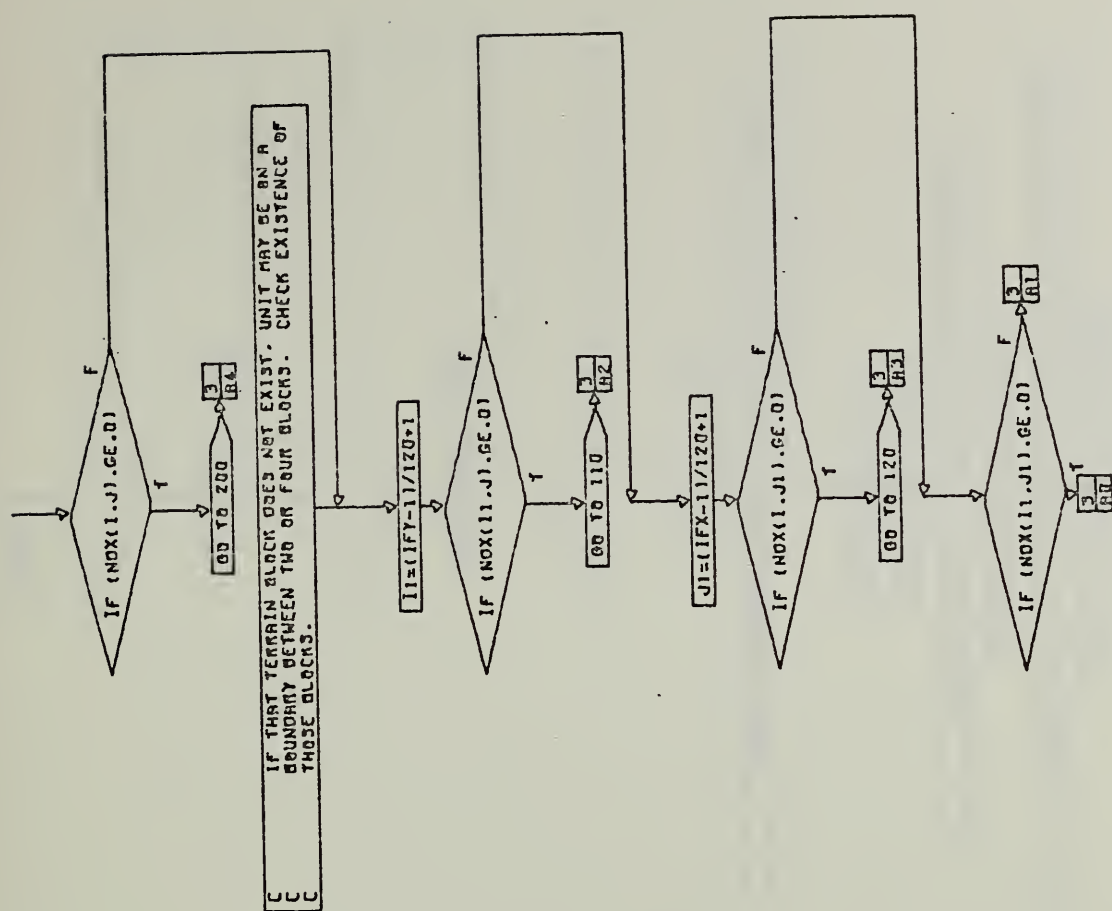
M=1

RETURN

COMPUTE THE SUBSCRIPTS FOR THE REQUIRED TERRAIN BLOCK.

100
 I=IFY/120+1

J=IFX/120+1



END

CO. 5. FINAL

